AMSTRAD

6128 plus

BIEN DEBUTER

EDITIONS MICRO APPLICATION



LIVRE DATA BECKER

SZCZEPANOWSKI

Bien Débuter



EDITIONS MICRO APPLICATION

Copyright © 1985 DATA Becker GmbH
Merowingerstraße, 30
4000 Düsseldorf

© 1990 Micro Application
58, rue du Faubourg Poissonnière
75010 Paris

Auteur Szczepanowski

Traducteur Pascal Hausmann

'Toute représentation ou reproduction, intégrale ou partielle, faite sans le consentement de MICRO APPLICATION est illicite (Loi du 11 Mars 1957, article 40, 1er alinéa).

Cette représentation ou reproduction illicite, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

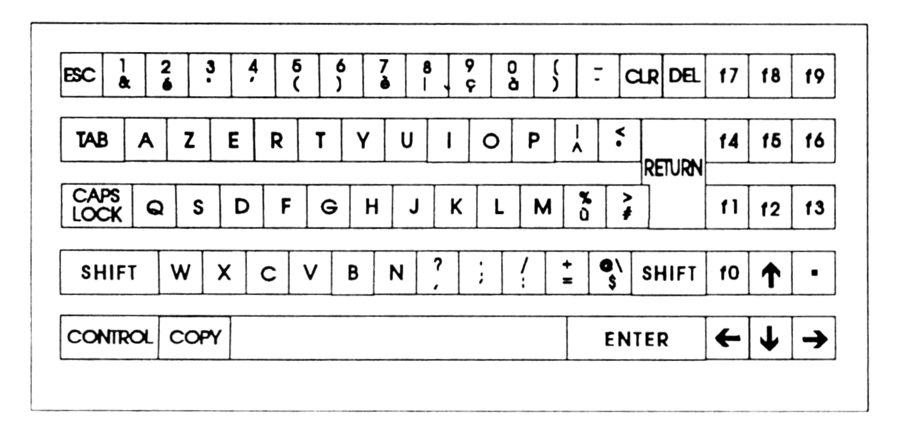
La Loi du 11 Mars 1957 n'autorise, aux termes des alinéas 2 et 3 de l'article 41, que les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à l'utilisation collective d'une part, et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration'.

ISBN : 2-86899-393-1

Collection dirigée par Philippe Olivier
Edition réalisée par Frédérique Beaudonnet

Sommaire

1. Le clavier



Ce chapitre vous permettra d'apprendre à manier les 74 touches de votre CPC 6128 Plus. Grâce à ces touches, vous pourrez utiliser pleinement toutes les possibilités que recèle votre ordinateur : entrée de textes, créations d'images ou graphiques, envoi de messages ou résolution de problèmes mathématiques, etc... Lorsque vous maîtriserez toutes les touches, le CPC 6128 Plus sera pour vous un partenaire obéissant qui vous aidera à résoudre toutes sortes de problèmes.

La maîtrise du clavier est bien sûr indispensable pour les fanas de la programmation qui veulent pouvoir explorer chaque bit de l'ordinateur mais elle est également indispensable pour ceux qui souhaitent utiliser leur ordinateur uniquement avec des programmes commerciaux. Elle leur permettra en effet de ne pas avoir à rechercher désespérément dans le manuel ce que peuvent vouloir dire des messages tels que "APPUYEZ SUR LA TOUCHE RETURN POUR FAIRE EDITER LE CONTENU DE L'ECRAN".

La connaissance du clavier est donc le minimum de savoir nécessaire pour tout utilisateur même très occasionnel d'un ordinateur.

Il n'est pas nécessaire toutefois de savoir taper à la machine pour utiliser un ordinateur. Le principal est de bien connaître le clavier ; vous verrez ensuite que vous trouverez les touches de plus en plus facilement et que vous taperez donc de plus en plus vite les instructions ou les lignes de programme.

Nous allons décrire dans ce chapitre de façon particulièrement détaillée les touches de couleur qui sont très importantes puisqu'elles permettent de traiter un texte, d'interrompre un programme et de transmettre des instructions à l'ordinateur.

Données générales sur le clavier

Au premier abord, le clavier du CPC 6128 Plus ressemble fort à un clavier classique de machine à écrire. Mais si vous regardez de plus près, vous pourrez constater la présence de quelques touches supplémentaires (touches de fonction), dont nous expliquerons plus tard d'utilisation.

Il vaut mieux que vous n'expérimentiez pas le fonctionnement du clavier avant de connaître la fonction des différentes touches. Bien sûr vous ne risquez pas de détruire l'ordinateur par des manipulations malhabiles, mais vous risquez de perdre du temps et n'être déçu par vos premières expériences.

En voiture !

```
 BASIC 1.1

Ready
■
```

Il est temps de réveiller les circuits en sommeil de votre CPC 6128 Plus. Insérez la cartouche du Basic puis allumez votre ordinateur. Il vous faut allumer à la fois le commutateur du moniteur et celui de l'ordinateur. Vous voyez alors apparaître un message indiquant que l'ordinateur est prêt à suivre vos instructions. L'ordinateur dispose d'une capacité mémoire de 128 kilo-octets soit 131072 cases mémoire. Sur ces 131072 cases

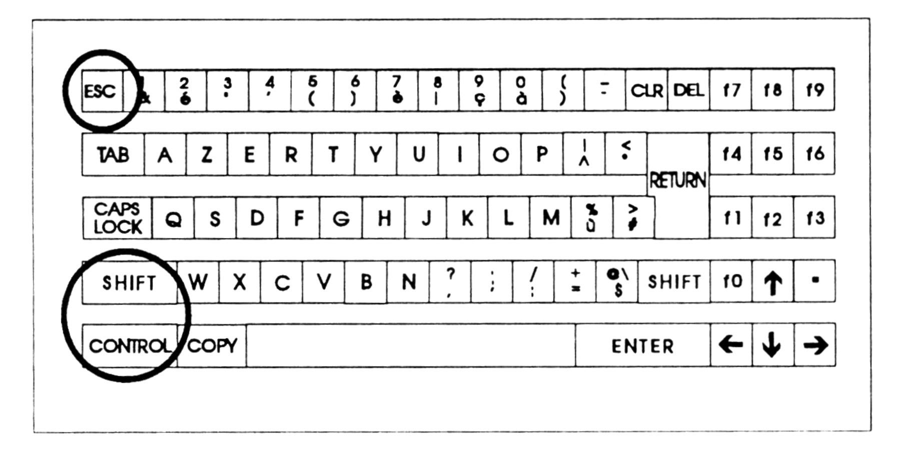
mémoire, vous pouvez en utiliser environ 40000 pour vos propres programmes Basic. Le reste des 64 premiers kilo-octets de la mémoire est utilisé par l'ordinateur pour le système d'exploitation et le langage de programmation Basic. Le deuxième bloc de 64 kilo-octets peut être utilisé comme mémoire de données.

C'est la société Locomotive Software Ltd. qui a réalisé le Basic intégré de votre ordinateur qui est d'ailleurs, comme vous le verrez bientôt, très puissant. Il s'agit de la version 1.1 de ce Basic. Si des améliorations devaient se révéler par la suite nécessaires, ce numéro serait modifié en conséquence. Appuyez sur la touche de fonction [F1] pour accéder au Basic, avant trente secondes, sinon le jeu Burnin' Rubber se chargera.

Le message Ready (prêt) à la dernière ligne indique que le système d'exploitation de votre ordinateur attend maintenant des instructions. Ce message indique donc que l'ordinateur est à votre disposition.

Venons-en maintenant au petit carré en dessous de Ready. Il vous permet de vous orienter sur l'écran. En effet, si vous voulez écrire quelque chose sur l'écran, vous devez savoir où sortira le caractère que vous allez entrer en frappant une touche. Ce point de repère est appelé "curseur".

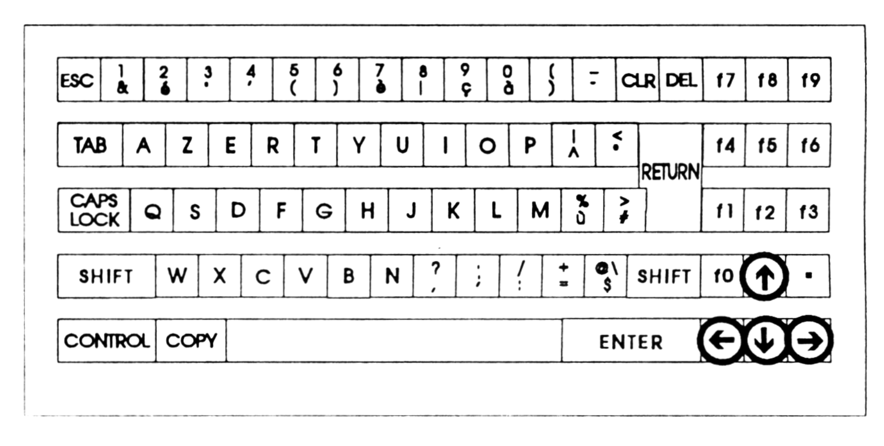
Le coup des trois doigts



Il arrive souvent qu'on ait besoin de replacer l'ordinateur dans son état initial, c'est-à-dire de provoquer ce qu'on appelle un RESET. Sur le CPC 6128 Plus, cela est très facile à réaliser : il suffit de tenir enfoncées simultanément les deux touches [Shift] et [Control] puis d'appuyer sur la touche [ESC]. Faites attention à ne relâcher les touches [Shift] et [Control] qu'après avoir appuyé sur la touche [ESC]. Vous voyez que le résultat est exactement identique à ce que vous obtenez lorsque vous allumez votre ordinateur.

Bien sûr, il vaut mieux y réfléchir à deux fois avant de provoquer un RESET car vous détruisez ainsi tout programme et toutes les données en mémoire.

Les touches curseur



Ces touches se trouvent au-dessus du pavé numérique, à la droite du clavier. Ces touches permettent de déplacer le curseur dans les directions correspondant aux flèches. Si vous appuyez par exemple 20 fois sur la touche portant une flèche dirigée vers la droite, vous verrez le curseur se déplacer de 20 cases vers la droite. Il se trouvera alors au beau milieu de l'écran.

```
BASIC 1.1

Ready
                    ■
```

Mais que se passe-t-il lorsque le curseur dépasse le bord droit d'une ligne de l'écran ? Essayez : appuyez à nouveau 20 fois sur la touche curseur DROITE. La vingtième fois que vous appuierez sur cette touche, le curseur apparaîtra dans la première colonne de la ligne suivante.

```
BASIC 1.1

Ready

■
```

Il est assez pénible d'appuyer 20 fois sur la touche curseur pour arriver au milieu de l'écran. Il existe un moyen plus simple : une fois que vous avez appuyé sur la touche curseur, vous pouvez la tenir enfoncée et le curseur se déplacera de lui-même vers la droite tant que vous tiendrez la même touche enfoncée. Il en va de même pour les autres touches. Essayez : vous voyez à quelle vitesse le curseur fonce vers la droite de l'écran.

Pour déplacer le curseur dans l'autre sens, utilisez la touche curseur [←]. Essayez également cette touche : vous voyez comme le curseur se déplace dans l'autre sens.

```
BASIC 1.1

Ready
                    ■
```

Comment pourrions-nous maintenant replacer le curseur dans son emplacement de départ, sous Ready ? Vous pourriez tenir enfoncée la touche curseur [←] jusqu'à ce que le curseur remonte toutes les lignes qui le séparent de Ready. Il y a cependant un

moyen plus rapide : les touches curseur [↑] et [↓] vous permettent de déplacer le curseur verticalement aussi rapidement que vous avez pu le déplacer jusqu'ici horizontalement avec les touches [→] et [←].

Amenez donc le curseur sur le bord gauche de l'écran. Appuyez trois fois sur la touche curseur [↓]. Vous voyez que le curseur est descendu de trois lignes.

```
 BASIC 1.1

Ready

■
```

Essayez maintenant d'amener le curseur dans la troisième ligne de l'écran. Utilisez pour cela la touche curseur [↑].

```
 BASIC 1.1

Ready
```

Ici encore, vous pouvez utiliser la fonction de répétition des touches : si vous tenez enfoncée la touche curseur [↑], vous voyez que le curseur se déplace automatiquement vers le haut, sans que vous ayez à appuyer à nouveau sur cette touche.

Pour vous habituer à manier les touches curseur qui vous seront très utiles plus tard, par exemple pour effectuer des modifications, essayez de tracer un carré invisible. Exercez-vous à amener le curseur dans n'importe quel endroit de l'écran. C'est ainsi que vous apprendrez à maîtriser le maniement des touches curseur.

Edition avec les touches curseur

Le terme d'édition est employé en informatique dans un sens particulier, assez différent du sens courant : éditer signifie

traiter des textes, c'est-à-dire créer et modifier un texte. On parle déjà d'édition lorsque vous écrivez quelques mots sur l'écran.

Sur certains ordinateurs, vous pouvez déplacer à volonté le curseur sur l'écran, entrer un texte, puis continuer à déplacer librement le curseur. Le CPC 6128 Plus présente à cet égard une importante particularité : vous pouvez déplacer librement le curseur jusqu'à ce que vous entriez un caractère quelconque au clavier.

A partir du moment où vous avez entré un ou plusieurs caractères, vous ne pouvez plus déplacer le curseur qu'à l'intérieur des caractères entrés. Assez de théorie! Voyons comment cela se passe en pratique. Replacez votre ordinateur dans son état initial ([Shift]-[Control]-[ESC]). Ecrivez maintenant le mot "ordinateur" sur l'écran, même si vous mettez un certain temps à trouver les bonnes touches :

```
 BASIC 1.1

Ready
Ordinateur█
```

Le curseur se trouve maintenant placé à la suite de la dernière lettre de ce mot et il refuse obstinément de prendre une autre direction que la gauche. Essayez toujours, même si vous êtes condamné à ne pas avoir beaucoup de succès. Le curseur ne vous permet plus que de traiter le texte que vous avez entré. c'est pourquoi vous ne pouvez quitter le texte que vous avez entré. Tenez par exemple la touche curseur [←] enfoncée, vous constaterez que le curseur s'arrête une fois arrivé au bord gauche de l'écran. Vous pouvez maintenant essayer dans l'autre sens, vous obtiendrez le même résultat : il semble qu'il y ait un mur après la dernière lettre.

Pourquoi cette limitation du mouvement du curseur ? Pour bien le comprendre, il faut se demander pourquoi on écrit à l'écran. Si vous entrez par exemple un programme, vous transmettez chaque ligne de programme à l'ordinateur avec la

touche [Return]. Vous ne pouvez donc pas quitter une ligne de programme tant que vous ne l'avez pas terminée, c'est-à-dire tant que vous ne l'avez pas validée avec la touche [Return].

C'est à ce moment seulement que le curseur retrouve sa liberté de mouvement.

Revenons au mot "ordinateur". Vous pouvez rajouter des lettres en n'importe quel endroit de ce mot. Il suffit pour cela de placer le curseur sur la lettre devant laquelle vous voulez ajouter une ou plusieurs autres lettres. On peut donc dire que le CPC 6128 Plus est en permanence en mode d'insertion, c'est-à-dire que vous pouvez insérer des caractères en permanence à l'endroit où se trouve le curseur. Transformons maintenant le mot "ordinateur" en "coordinateur". Amenez le curseur sur le bord gauche de l'écran.

```
 BASIC 1.1

Ready
■rdinateur
```

Le curseur se trouve maintenant sur la lettre "o" devant laquelle nous voulons insérer "co". Appuyez simplement sur la touche "c" puis sur la touche "o". Vous voyez que chaque lettre repousse le texte situé à la droite du curseur d'une case vers la droite.

```
 BASIC 1.1

Ready
co■rdinateur
```

Supposons que nous voulions maintenant (au mépris de l'orthographe!) ajouter encore un trait d'union entre "coordina" et "teur". Vous savez certainement maintenant comment vous devez procéder : vous devez placer le curseur sur le "t" puis appuyer sur la touche "-", à la droite du zéro.

```
 BASIC 1.1

Ready
Coordina-█eur
```

Si vous replacez le curseur à l'extrême droite du mot, vous pouvez poursuivre l'entrée du texte. Rappelons que toute entrée doit se terminer avec la touche [Return]. Si vous appuyez toutefois sur la touche [Return] après avoir entré le texte de notre exemple, le CPC 6128 Plus va certainement grogner.

Il va en effet vous envoyer le message 'Syntax error' qui signifie 'Je ne comprends pas le message que tu m'envoies.' Evidemment : la seule langue que comprenne votre ordinateur est le Basic et il faudra d'abord que vous l'appreniez avant que vous puissiez travailler avec votre CPC 6128 Plus. Mais patientez encore quelques pages.

Essayez pour le moment de vous exercer à insérer des lettres dans un texte quelconque, jusqu'à ce que vous possédiez parfaitement le mode d'insertion.

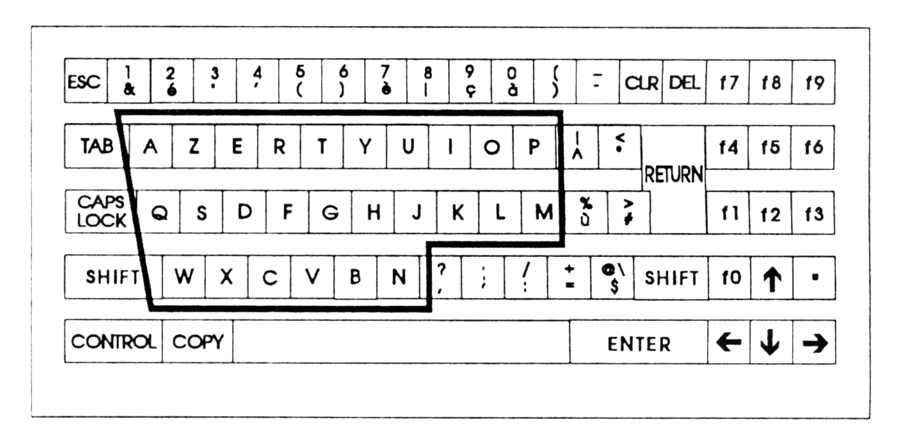
Vidage de l'écran

Lorsque vous constatez que tout ce qui figure à l'écran est sans intérêt et que vous voulez commencer autre chose, sachez que vous pouvez nettoyer l'écran très rapidement. Pas bien sûr avec un chiffon qui peut enlever la poussière à la surface de l'écran mais plutôt les caractères qui sont stockés dans la mémoire de l'ordinateur. Pour un ordinateur qui peut traiter environ un million d'instructions par seconde, c'est bien sûr un jeu d'enfant que de vider en un instant l'écran des quelques 1000 caractères qui peuvent y figurer. Sur beaucoup d'autres ordinateurs vous pouvez donner l'ordre de vider l'écran en appuyant sur une seule touche.

Sur le CPC 6128 Plus, ce n'est pas si simple puisqu'il faut tout de même utiliser 4 touches pour vider l'écran. Vous devez en

effet entrer l'instruction CLS qui, comme toutes les instructions, doit être terminée par l'emploi de la touche [Return] qui transmet l'instruction à l'ordinateur. CLS vient de l'anglais comme toutes les autres instructions que comprend votre ordinateur : CLS signifie CLear Screen : nettoyer l'écran. Entrez donc CLS puis appuyez sur la touche [Return] : l'écran se vide instantanément et vous voyez apparaître le message Ready qui indique à nouveau que l'ordinateur attend de nouvelles instructions de votre part. Notez bien cette instruction que nous aurons encore souvent à utiliser dans le cours de cet ouvrage.

Les touches avec des lettres



Comme nous l'avons déjà indiqué, la disposition de ces touches est très peu différente de celle d'une machine à écrire. Chaque fois que vous appuyez sur une de ces touches, apparaît la lettre minuscule correspondante. Avant que nous n'utilisions ces touches, videz l'écran avec l'instruction CLS.

Nous avons maintenant un écran "propre" pour y faire notre travail d'édition. Ne pensez pas surtout que vous pouvez maintenant écrire de longs textes et que votre ordinateur va tout retenir et mettre en mémoire au fur et à mesure. Les choses sont plus compliquées et ce que vous pouvez écrire maintenant directement est simplement placé dans une mémoire provisoire très fugitive. Vous apprendrez dans l'introduction au Basic

comment on peut sauvegarder des textes après les avoir entrés au clavier.

Ecrivez maintenant toutes les lettres sur l'écran en suivant l'ordre alphabétique. Vous constaterez qu'il est très difficile de trouver au départ telle ou telle touche quand on n'est pas précisément une dactylo. Vous voyez maintenant à l'écran toutes les lettres minuscules dont dispose votre ordinateur.

```
 BASIC 1.1

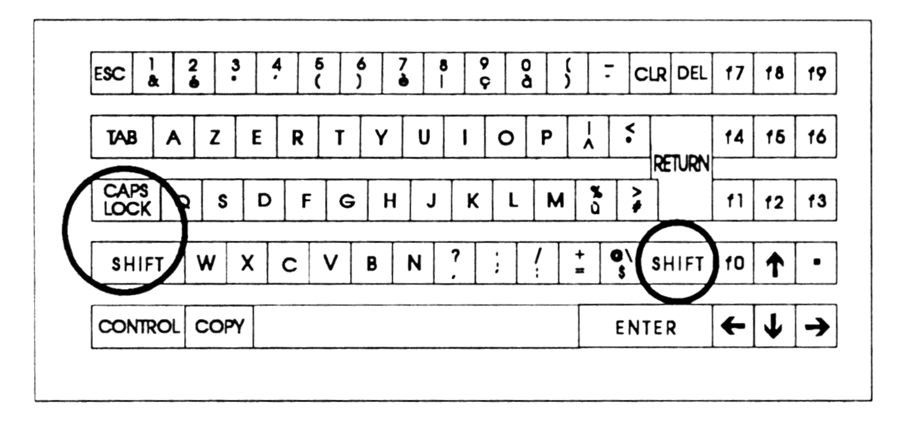
Ready
abcdefghijklmnopqrstuvwxyz■
```

Videz à nouveau l'écran et entrez à nouveau toutes les lettres. Cette fois-ci, ne suivez pas l'ordre alphabétique et entrez les lettres dans l'ordre où elles figurent sur le clavier. Vous remarquerez qu'il est beaucoup plus facile d'entrer les lettres dans cet ordre.

```
 BASIC 1.1

Ready
azertyuiopqsdfghjklmwxcvbn■
```

Majuscules/Minuscules



Vous avez déjà remarqué que toutes les lettres que vous entrez apparaissent à l'écran en minuscule. Il est cependant souvent nécessaire de pouvoir produire également des lettres majuscules, notamment lorsqu'on fait du traitement de texte. Sur les machines à écrire, une touche spéciale permet d'obtenir des majuscules. Il en va de même sur le CPC 6128 Plus : la touche [Shift] qui existe en deux exemplaires vous permet de sortir des majuscules. Vous devez la tenir enfoncée pendant que vous appuyez sur la touche correspondant à la majuscule que vous voulez sortir. Essayez par exemple d'écrire à l'écran le nom "Charles Lefevre".

```
 BASIC 1.1

Ready
Charles Lefevre█
```

Nous avons déjà écrit toutes les lettres minuscules sur l'écran. Essayez maintenant d'écrire toutes les lettres majuscules dans l'ordre alphabétique. Pour que vous ne soyez pas obligé de tenir en permanence la touche [Shift] enfoncée, vous disposez d'une autre touche de fonction, [CapsLock]. Si vous appuyez une fois sur cette touche, toutes les lettres que vous entrerez ensuite apparaîtront en majuscules.

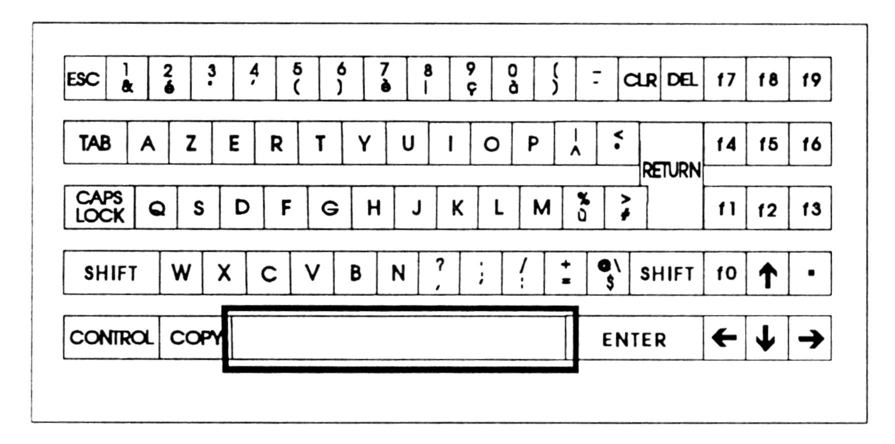
```
 BASIC 1.1

Ready
ABCDEFGHIJKLMNOPQRSTUVWXYZ█
```

Si vous appuyez maintenant une seconde fois sur la touche [CapsLock], vous pouvez à nouveau écrire en minuscules.

Pour écrire les chiffres situés sur la première rangée (0 à 9), il faut appuyer sur [CapsLock] et [Control] en même temps.

La touche espace



Cette touche se trouve également sur toutes les machines à écrire. Elle vous permet de placer des espaces entre les mots. Videz l'écran et entrez la phrase suivante :

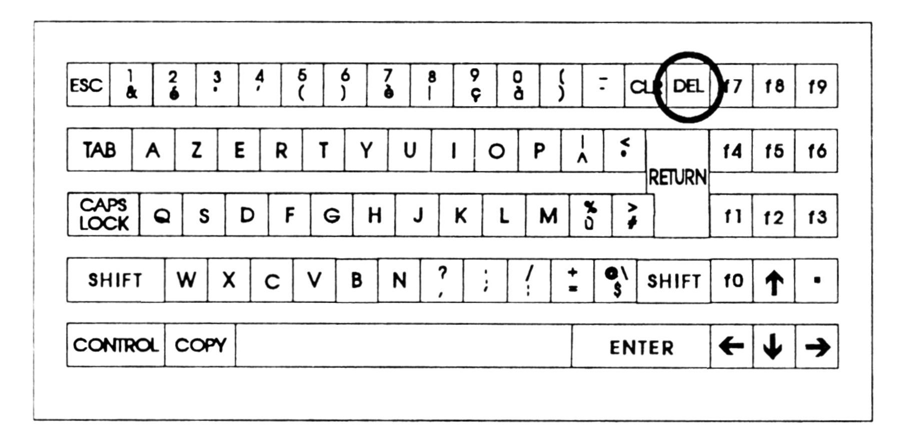
```
A chaque jour suffit sa peine
```

Il vous faut appuyer sur la touche espace entre les différents mots de cette phrase.

```
 BASIC 1.1

Ready
A chaque jour suffit sa peine█
```

La touche DEL



[Del] est mis pour le mot anglais DELete qui signifie supprimer. La touche [Del] vous permet de supprimer le caractère placé à la gauche du curseur. Cette touche est donc indispensable pour corriger les fautes de frappe : lorsque vous venez de taper sur la mauvaise touche, vous n'avez qu'à appuyer sur la touche [Del] et le caractère erroné disparaît.

Videz maintenant l'écran et entrez la phrase suivante :

```
Les fautes de frappe ne pardonnent pah
```

Visiblement, il s'est produit une faute de frappe lors de l'entrée de la dernière lettre.

```
BASIC 1.1

Ready
Les fautes de frappe ne pardonnent pah█
```

Pas besoin de Tipp-Ex. Il suffit de frapper une touche pour supprimer la dernière lettre entrée. Appuyez donc sur la touche [Del] :

```
 BASIC 1.1

Ready
Les fautes de frappe ne pardonnent pa■
```

Vous pouvez maintenant écrire ici la lettre qui convient et la faute de frappe est définitivement oubliée.

```
 BASIC 1.1

Ready
Les fautes de frappe ne pardonnent pas■
```

Mais cet exemple ne vous permet pas encore de mesurer pleinement tout l'avantage que présente cette touche. Essayons donc un autre exemple. Videz l'écran puis entrez la phrase suivante :

```
Les fautes de frappe ne parrdonnent pas
```

Ici aussi une faute de frappe s'est glissée dans notre texte puisque les deux 'r' de parrdonnent n'ont pas lieu d'être.

```
 BASIC 1.1

Ready
Les fautes de frappe ne parrdonnent pas■
```

Il n'est pas nécessaire dans ce cas de réécrire le dernier mot entré à partir de la faute de frappe. En effet, si vous supprimez un caractère avec la touche [Del], tous les caractères placés sur cette ligne à la droite du curseur sont automatiquement ramenés vers la gauche. Comme le principe de fonctionnement de cette touche est plus difficile à expliquer qu'à constater, faites-en vous-même l'expérience : amenez le curseur à la suite du 'r' à supprimer. Utilisez pour cela la touche curseur [←].

```
 BASIC 1.1

Ready
Les fautes de frappe ne pa█donnent pas
```

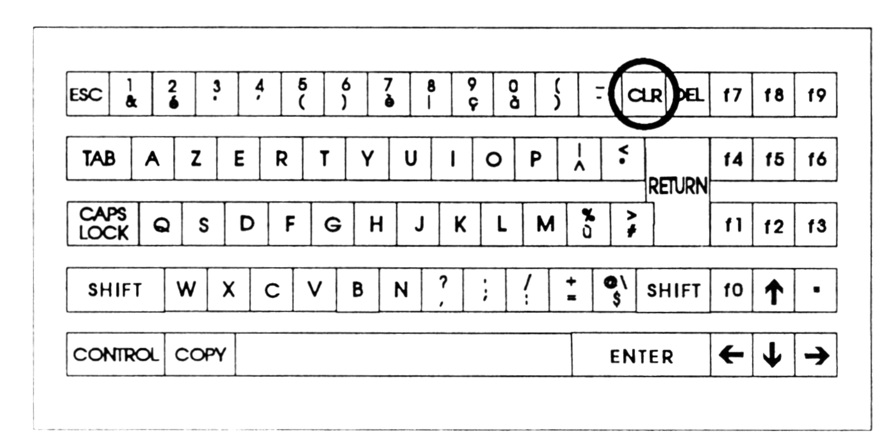
Si vous appuyez maintenant sur la touche [Del], vous constatez que le 'r' à gauche du curseur est supprimé et que le texte placé à la suite sur la ligne, y compris le caractère sous le curseur, est ramené vers la gauche.

Si vous replacez maintenant le curseur à l'extrême droite du texte, vous pouvez continuer à écrire. Il vous suffit pour cela de tenir enfoncée la touche curseur DROITE jusqu'à ce que le curseur se trouve placé à l'extrême droite de la phrase.

La touche CLR

Alors que la touche [Del] supprime le caractère à gauche du curseur et ramène le texte suivant ce caractère vers la gauche, la touche [CLR] est également d'une grande aide pour faciliter l'édition à l'écran : elle supprime le caractère SOUS le curseur et ramène le texte à droite du curseur vers la gauche. Prenons encore un exemple. Entrez encore une fois une phrase erronée :

```
Il fait très beau
```



Vous constatez en fait qu'il ne fait pas si beau que ça. Il faut donc supprimer le mot "très". Bien entendu, vous pourriez le faire également avec la touche [Del] mais nous voulons justement apprendre à utiliser la touche [CLR]. Comme [CLR] supprime le caractère sous le curseur, c'est sur le "t" de "très" qu'il faut amener le curseur.

```
 BASIC 1.1

Ready
Il fait █rès beau
```

Si vous appuyez maintenant quatre fois sur la touche [CLR], le mot 'très' sera supprimé.

```
 BASIC 1.1

Ready
Il fait beau
```

Les touches [Del] et [CLR] sont deux touches de fonction extrêmement pratiques pour vous faciliter le travail d'édition lors de l'écriture de programmes ou de l'entrée de textes quelconques.

Densité d'écriture

Vous avez certainement déjà constaté que chaque ligne de l'écran comporte 40 caractères. Vous pouvez ainsi écrire 1000 caractères sur les 25 lignes de l'écran. Sur la plupart des ordinateurs de cette classe, on considère généralement que c'est insuffisant et beaucoup d'utilisateurs essayent de se procurer, parfois à grands frais, des extensions leur permettant d'obtenir un écran de 80 colonnes.

Les concepteurs du CPC 6128 Plus ont donc été très inspirés lorsqu'ils ont décidé que leur ordinateur disposerait d'une densité d'écriture variable. L'utilisateur peut en effet choisir s'il

veut avoir 20, 40 ou même 80 caractères par ligne. Ce choix se fait d'ailleurs de la façon la plus simple, avec une instruction Basic. Nous connaissons déjà une instruction Basic, CLS. Cette instruction "nettoie" l'écran en le vidant.

L'instruction de choix de la densité d'écriture est MODE. Comme il y a trois possibilités de choix, cette instruction doit être suivie d'un chiffre indiquant quelle possibilité vous sélectionnez. Il y a donc trois instructions MODE différentes :

MODE 0 qui fixe le nombre de colonnes à 20
MODE 1 qui fixe le nombre de colonnes à 40
MODE 2 qui fixe le nombre de colonnes à 80

Vous pouvez essayer tout de suite l'effet de ces instructions. Entrez d'abord MODE 0. Vous devez terminer cette instruction, comme toutes les autres, en appuyant sur la touche [Return] pour transmettre l'instruction à l'ordinateur. Tant que vous n'appuyez pas sur la touche [Return], vous attendrez en vain que l'ordinateur réagisse.

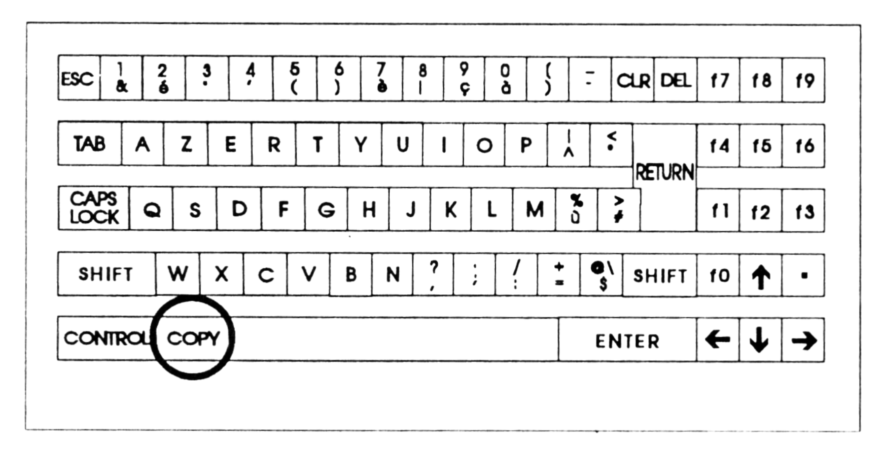
En voyant le mot Ready écrit en aussi grosses lettres, vous vous dites certainement que ce mode convient aux mal-voyants! Bien sûr, les lettres en mode 0 sont beaucoup plus faciles à lire que lorsqu'il y a 40 caractères par ligne, comme c'était le cas jusqu'ici. En fait ce mode servira essentiellement dans les jeux. Si vous vous imaginez que vous pouvez utiliser ce mode pour écrire un titre en grosses lettres et que vous pourrez écrire en-dessous des lignes en écriture normale, nous devons malheureusement vous ôter vos illusions. Vous ne pouvez écrire à l'écran que dans un seul mode car l'instruction MODE a pour premier effet de vider l'écran de son contenu.

Mais vous pouvez également choisir le mode avec 80 caractères par ligne. Entrez pour cela l'instruction MODE 2 et vous constatez que les caractères sont maintenant beaucoup plus petits. Vous disposez d'un écran en 80 colonnes!

Ecrivez un texte quelconque sur l'écran et vous constaterez que les petites lettres sont étonnamment lisibles.

Il est toutefois certain que la lisibilité est très supérieure, en 80 colonnes, avec un moniteur monochrome. Par ailleurs le CPC 6128 Plus dispose de si belles couleurs qu'il serait dommage de s'en priver. Cela ne signifie pas bien sûr qu'il faille avoir deux moniteurs, l'un monochrome et l'autre couleur.

Le curseur de copie



Tout ordinateur vous permet de conserver des textes, des programmes ou des données et de les modifier ensuite à tout moment. On n'écrit en effet jamais un programme de but en blanc et on a souvent au contraire à le modifier. Pour que vous puissiez par la suite avoir plus de facilité à écrire et modifier vos programmes, voyons comment on peut modifier une suite de caractères déjà entrés.

Prenons un exemple. Supposons que vous vouliez entrer l'instruction CLS et que vous fassiez une faute de frappe en entrant 'cld' au lieu de 'cls' :

```
Ready
cld
Syntax error
Ready
■
```

Une fois entrée cette instruction, le message Syntax error indique qu'il s'agit d'une instruction incorrecte. Nous voulons cependant éviter d'avoir à retaper cette instruction. Sur d'autres ordinateurs, l'éditeur d'écran vous permet de ramener le curseur sur la ligne où figure l'instruction erronée, de corriger cette instruction puis de la valider avec [Return]. Cela n'est pas possible sur le CPC 6128 Plus. Essayez tout de même. Amenez le curseur sur le 'd' de cld puis remplacez-le par un 's'. Vous voyez maintenant, à gauche du curseur, l'instruction correcte cls. Si vous appuyez maintenant sur la touche [Return] pour valider cette instruction, vous voyez que l'écran n'est pas vidé et que l'ordinateur vous envoie à nouveau le même message d'erreur.

```
Ready
cls
Syntax error
Ready
■
```

En effet, sur le CPC 6128 Plus, il faut entrer toutes les lettres d'une instruction avant d'utiliser la touche [Return]. Les caractères qui figurent sur l'écran ne peuvent plus être complétés ou modifiés une fois que vous avez appuyé sur la touche [Return]. Lorsqu'on a affaire à des lignes d'instructions assez longues, il est bien sûr très ennuyeux de devoir entrer à nouveau la totalité de la ligne lorsqu'on a commis la moindre erreur. Heureusement, le CPC 6128 Plus vous évite ce travail fastidieux. Vous pouvez en effet recopier les caractères de "l'ancienne" ligne dans un autre emplacement de l'écran. Il faut un peu d'exercice pour maîtriser cette méthode. Nous allons vous apprendre cette technique pas à pas. Videz donc l'écran (avec l'instruction correcte cette fois) et entrez à nouveau l'instruction erronée 'cld'.

1ère Etape - Placer le CURSEUR DE COPIE au début de la "nouvelle" ligne

Décidez d'abord où vous voulez copier "l'ancienne" ligne d'instructions et déplacez le curseur vers le début de cette ligne. Dans notre exemple, ce sera la cinquième ligne de l'écran.

```
Ready
cld
Syntax error
Ready

■
```

2ème étape - Placer le CURSEUR au début de "l'ancienne" ligne

Vous maîtrisez déjà le curseur que nous venons de décrire.

Mais qu'est-ce alors que le curseur de copie, et comment peut-on le déplacer ? Le curseur de copie est un second curseur qui a la même apparence que le premier curseur. On le déplace également avec les touches curseur, mais en tenant simultanément enfoncée une touche [Shift]. Tenez donc enfoncée la touche [Shift] et actionnez la touche curseur [↑], jusqu'à ce que le curseur de copie ait atteint "l'ancienne" ligne.

```
Ready
■ld
Syntax error
Ready

■
```

3ème étape - copier les caractères voulus avec la touche COPY

La touche [COPY] déplace le curseur de copie ainsi que le curseur normal vers la droite et copie en même temps les caractères sous le curseur de copie. Cela semble plus compliqué que ça ne l'est dans la pratique. Essayez vous-même, en appuyant deux fois sur la touche [COPY].

```
Ready
cl■
Syntax error
Ready

cl■
```

Vous avez maintenant copié les deux premiers caractères de l'instruction CLS. Il ne faut pas copier la troisième lettre puisqu'il s'agit de la corriger. Appuyez donc sur la touche 's'.

```
Ready
cl■
Syntax error
Ready

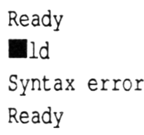
cls■
```

L'ancienne instruction a donc maintenant été copiée et modifiée. Appuyez maintenant sur la touche [Return] et l'instruction sera exécutée. La touche [Return] fait également disparaître le curseur de copie.

Il existe cependant une autre méthode de copie et de modification de lignes d'instructions. Nous avons copié et modifié "l'ancienne" ligne dans un autre emplacement de l'écran. L'autre possibilité consiste à modifier "l'ancienne" ligne à l'endroit même où elle se trouve. Nous avons échoué lorsque nous avons voulu amener tout simplement le curseur sur l'emplacement où devait être effectuée la correction. Essayons donc maintenant la seconde méthode, que nous vous présenterons également pas à pas. Entrer d'abord à nouveau l'instruction erronée 'cld'.

1ère étape - Placer le curseur au début de "l'ancienne" ligne.

Nous ne plaçons pas cette fois le curseur dans un autre endroit de l'écran, mais au début de la ligne d'instructions à modifier.



```
Ready
■ld
Syntax error
Ready
```

2ème étape - Placer le curseur de copie sur le premier caractère incorrect.

Appuyez maintenant deux fois sur la touche [COPY]. Les deux premiers caractères seront alors recopiés à l'endroit même où ils se trouvaient déjà puisque les positions des curseurs normal et de copie sont identiques.

C'est donc sur la touche COPY et non sur la touche curseur [→] que vous devez appuyer.

```
Ready
cl■
Syntax error
Ready
```

Cette fois aussi, il ne faut pas copier le troisième caractère mais le remplacer par un 's'. Appuyez donc sur la touche 's'.

```
Ready
cls■
Syntax error
Ready
```

Vous pouvez maintenant faire exécuter cette instruction corrigée en appuyant sur la touche [Return].

Vous connaissez donc maintenant deux méthodes pour modifier des lignes d'instructions à l'écran, la seconde semblant être la plus simple. A vous de décider, cas par cas, de celle que vous préférez utiliser.

Vous avez peut-être encore quelques difficultés à utiliser le curseur de copie. C'est pourquoi nous vous proposons un autre

exemple plus complet. Videz donc l'écran et entrez la phrase suivante, puis appuyez sur [Return] :

```
Ready
Il est de copier facile
Syntax error
Ready
■
```

A partir de ces mots nous allons reconstituer avec le curseur de copie la phrase "Il est facile de copier". Bien entendu, comme cette phrase ne constitue pas une instruction Basic, nous ne devons pas nous étonner d'obtenir le message "Syntax error" après avoir appuyé sur la touche [Return]. Mais cela ne prête pas à conséquence.

Amenons le curseur de copie au début de la ligne 6 où nous voulons voir apparaître la phrase corrigée.

```
Ready
Il est de copier facile
Syntax error
Ready

■
```

Il nous faut maintenant amener le curseur de copie au début du premier mot à copier. Tenez pour cela la touche [Shift] enfoncée et déplacez le curseur de copie avec les touches curseur.

```
Ready
■l est de copier facile
Syntax error
Ready

■
```

Nous pouvons maintenant copier le premier mot. Appuyez donc sur la touche [COPY], jusqu'à ce que le curseur de copie

soit sur le 'd'. L'espace entre 'est' et 'de' doit en effet être copié également.

```
Ready
Il est ■e copier facile
Syntax error
Ready

Il est ■
```

Il faut maintenant copier le second mot. Il nous faut donc amener à nouveau le curseur de copie au début du mot 'facile'. Tenez donc la touche [Shift] enfoncée et appuyez dix fois sur la touche curseur [→].

```
Ready
Il est de copier ■acile
Syntax error
Ready

Il est ■
```

Copiez maintenant ce mot et un espace en appuyant 7 fois sur la touche [COPY].

```
Ready
Il est de copier facile ■
Syntax error
Ready

Il est facile ■
```

Nous vous laissons maintenant le soin de trouver par vous-même comment copier 'de copier'; il faut simplement déplacer le curseur de copie puis appuyer sur la touche [COPY] ! Vous devez obtenir à l'écran l'image suivante :

```
Ready
Il est de copier■facile
Syntax error
Ready
```

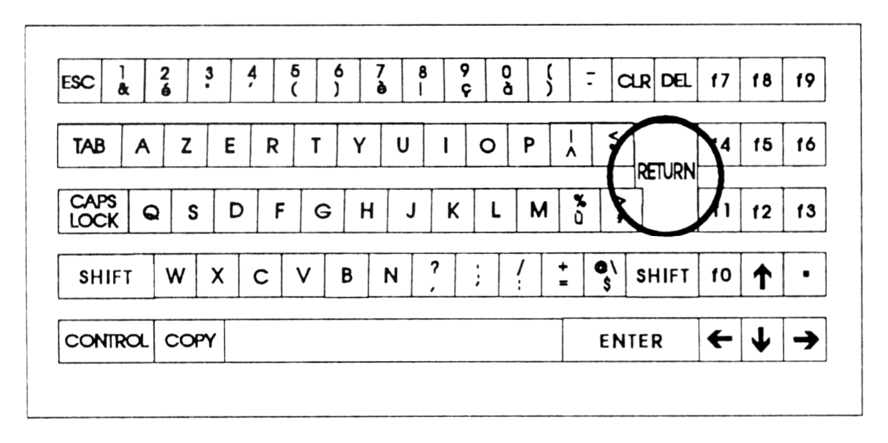
```
Il est facile de copier■
```

Vous y êtes arrivé ? Félicitations! Si vous faites encore quelques petits exercices vous maîtriserez vite cette technique pleinement. Il faut simplement faire attention à ne pas confondre les déplacements des deux curseurs entre eux. Sinon vous risquez de vous placer dans des situations dont vous n'arriverez à vous tirer qu'avec un peu plus d'expérience.

2. La première instruction

Pour expliquer l'utilisation des autres touches du clavier, il ne suffit plus d'écrire n'importe quels caractères sur l'écran. La plupart de nos lecteurs n'ont d'ailleurs certainement pas acheté le CPC 6128 Plus uniquement pour y écrire des textes à l'écran. C'est pourquoi nous allons maintenant utiliser l'ordinateur pour ce pour quoi il est fait, c'est-à-dire pour exécuter des instructions.

La touche Return



La touche [Return] est la touche la plus importante de l'ordinateur. Lorsque vous appuyez sur cette touche, le texte que vous avez entré est transmis à l'ordinateur. L'ordinateur interprète cette ligne comme une instruction et accomplit donc les tâches nécessaires. Une fois qu'il a exécuté une ligne d'instructions, l'ordinateur se présente à nouveau avec le message Ready. Il est donc à nouveau prêt à recevoir de nouvelles instructions. Essayez donc de saluer l'ordinateur en entrant "Bonjour" puis en appuyant sur la touche [Return]. Que croyez-vous que l'ordinateur répondra ? Mais voyez plutôt par vous-même.

```
Ready
bonjour
```

```
Syntax error
Ready
■
```

Vous voyez ici le premier message d'erreur du CPC 6128 Plus, Syntax error (erreur de syntaxe). La syntaxe, c'est-à-dire l'assemblage des caractères, n'est pas comprise par l'ordinateur.

Il existe aujourd'hui des ordinateurs capables de "discuter" avec des hommes, mais cela nécessite des ordinateurs cent mille fois plus chers et une mémoire cent mille fois plus puissante. Votre ordinateur par contre ne comprend que le Basic. Ce que vous entrez au clavier et transmettez en appuyant sur la touche [Return] est analysé par l'interpréteur Basic puis exécuté si votre entrée était correcte.

Votre CPC 6128 Plus est muni du langage de programmation Basic et ne comprend donc que les instructions de ce langage. Le Basic est le langage de programmation le plus répandu et celui qui place le débutant dans les meilleures conditions pour programmer. Les mots-instructions sont tirés de l'anglais. En combien de temps pourrez-vous apprendre ce langage ? Il n'est pas si simple de répondre. Bien entendu la réceptivité du débutant joue un rôle important mais également le temps que vous pourrez consacrer à votre ordinateur. Certaines personnes passent tous leurs loisirs devant l'ordinateur, si bien qu'il n'est plus rare de voir des annonces telles que "vends mon ordinateur pour éviter divorce". L'idéal est de consacrer suffisamment de temps à l'ordinateur pour pouvoir avoir la joie d'obtenir quelques résultats.

Avec quelques notions élémentaires de logique et d'algèbre, vous devriez pouvoir obtenir de premiers succès en trois mois à raison d'environ une dizaine d'heures par semaine. Bien sûr la capacité d'abstraction et la logique sont très utiles pour apprendre la programmation mais d'autre part, si vous n'en êtes pas richement pourvu, vous les verrez progresser de toute façon de programme en programme. En tout cas, pour pouvoir réaliser un programme d'envergure, tel qu'un programme de traitement de texte ou un programme de gestion de fichier, l'expérience de la programmation est essentielle. En effet la

qualité du style de programmation augmente de programme en programme. Sachez par ailleurs que beaucoup de possesseurs de microordinateurs arrivent déjà à programmer de petits jeux trois mois après avoir fait l'acquisition d'un ordinateur. Le principal est de conserver l'enthousiasme et d'avoir toujours autant de plaisir à travailler avec un ordinateur.

L'instruction Print

Cette instruction figure certainement dans tous les programmes. C'est elle qui vous permet en effet de sortir sur écran ou sur imprimante, ou d'enregistrer sur disquette des textes ou des résultats de calculs.

Nous ne nous intéressons pour le moment qu'au mode direct et nous n'écrivons donc pas encore de programme. Le mode direct signifie que vous entrez une instruction, que vous la terminez en appuyant sur la touche `Return` et que vous attendez un résultat immédiat. Cette fois, nous allons essayer de ne pas provoquer de message d'erreur. Entrez donc l'instruction 'print 10', suivie de `Return`.

```
Ready
Print 10
 10
Ready
■
```

L'instruction PRINT envoie donc sur l'écran tout ce qui est ajouté, comme paramètre, à l'instruction (ici 10). La sortie, sur d'autres périphériques que le moniteur, se fait avec une autre forme de PRINT. Les paramètres sont des expressions placées à la suite d'une instruction pour préciser les effets de cette instruction.

Ici 10 était le paramètre de la sortie sur écran. 'print 10' signifie donc 'affiche le nombre 10 sur l'écran'. Bien entendu vous ne pouvez pas seulement sortir des nombres, mais également n'importe quels autres caractères. Nous verrons dans ce chapitre que cette instruction est très polyvalente.

Calculer avec Print

Si vous avez des calculs à effectuer, vous pouvez parfaitement les faire sur votre CPC 6128 Plus. Ce serait tout de même extraordinaire s'il n'était pas possible de faire des calculs sur un ordinateur alors qu'un ordinateur coûte 80 fois le prix d'une calculatrice électronique !

Venons-en au fait et intéressons-nous d'abord aux quatre opérations fondamentales auxquelles correspondent sur le clavier de votre ordinateur les quatre symboles suivants :

+	pour l'addition
-	pour la soustraction
*	pour la multiplication
/	pour la division

Toutefois, vous ne pouvez pas faire de calcul sur un ordinateur de la même façon que sur une calculatrice. Vous ne pouvez pas entrer par exemple '12*2=' puisque votre ordinateur ne peut rien faire tant que vous n'entrez pas une instruction. Pour ne pas frustrer les lecteurs qui aiment bien tout expérimenter par eux mêmes, nous vous invitons d'ailleurs à entrer '12*2' puis à transmettre cette "instruction" en appuyant sur la touche [Return].

```
Ready
12*2=
■
```

Que s'est-il passé ? Le curseur est simplement passé au début de la ligne suivante et l'ordinateur n'a affiché ni Syntax error ni Ready et il n'a pas non plus exécuté votre "instruction". Si nous voulions expliquer exactement ce qui s'est passé, cela nous entraînerait dans des explications qui viendront seulement plus tard, lorsque nous expliquerons ce que c'est qu'un programme. Disons simplement pour le moment que toute ligne d'instructions qui commence par un nombre quelconque est considérée par l'ordinateur comme une ligne de programme. C'est même là la seule différence qui permet à l'ordinateur de distinguer le mode direct du mode programme : une ligne

d'instructions ne commençant pas par un nombre est en mode direct et l'ordinateur l'exécute immédiatement ; une ligne d'instructions commençant par contre par un nombre est en mode programme et l'ordinateur ne l'exécutera que lorsque vous lui aurez demandé d'exécuter le programme. Dans notre exemple, l'ordinateur considère donc que vous avez entré une ligne de programme portant le numéro 12 dont le texte est '*2='. Quelle est donc l'instruction correcte pour faire calculer et sortir sur écran le résultat de 12*2 ? La solution est simple : il faut entrer 'print 12*2'. Attention : chaque fois que vous entrez une instruction, vous devez la terminer en appuyant sur [Return]. En effet, ce que vous écrivez sur l'écran n'intéresse pas l'ordinateur. Seul l'intéresse ce que vous lui transmettez avec la touche [Return].

```
Ready
Print 12*2
 24
Ready█
```

Enfin ! Pour la première fois vous constatez que votre ordinateur a travaillé pour vous. Il a suivi fidèlement votre instruction, il a multiplié 12 par 2 et il a sorti le résultat sur l'écran.

Remarquons à ce propos que les langages de programmation ne paraissent compliqués que parce que vous êtes obligé d'abréger considérablement les instructions que vous donnez à l'ordinateur. Un langage de programmation doit en effet avant tout permettre d'éviter toute confusion. Il n'y a pas de langage de programmation et il n'y en aura certainement jamais qui vous permette d'envoyer des instructions telles que 'CALCULE 2*12 ET AFFICHE LE RESULTAT'. Le Basic est une convention précise qui dispose de beaucoup moins de mots et de synonymes que le langage courant. C'est pourquoi il faut apprendre ce langage progressivement.

Contrairement à ce que permet le Basic d'autres ordinateurs, il est absolument indispensable de séparer l'instruction print de son paramètre par un espace. Sinon, vous obtenez un Syntax error. D'une manière générale, toutes les instructions de votre

CPC 6128 Plus doivent être séparées d'un paramètre par un espace. Mais revenons au calcul. Calculez successivement 12+2, 12-2, 12*2 puis 12/2. Vous devriez obtenir à l'écran l'image suivante :

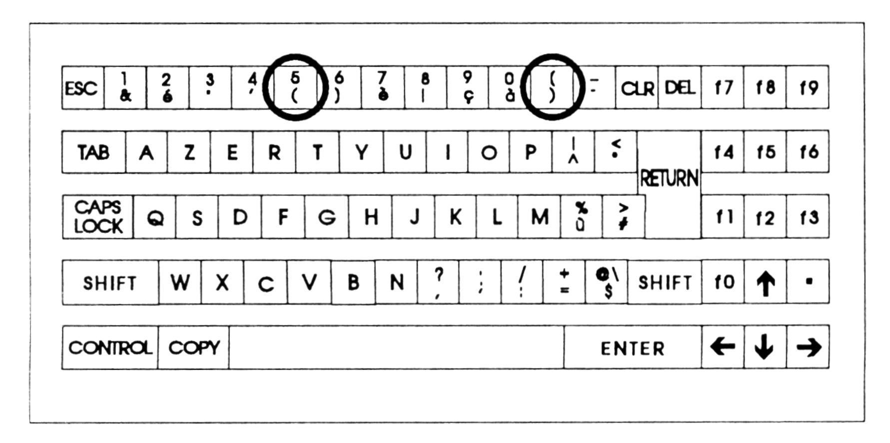
```
Ready
Print 12+2
 14
Ready
Print 12-2
 10
Ready
Print 12*2
 24
Ready
Print 12/2
 6
Ready
■
```

C'est ce que vous obtenez ? Bien! Si vous n'y êtes pas arrivé, relisez le paragraphe précédent attentivement.

A côté de calculs aussi simples, vous pouvez également faire effectuer par l'ordinateur des calculs comprenant jusqu'à 255 caractères. Dans ce cas, vous devez tenir compte de la hiérarchie des différents types d'opérations qui est d'ailleurs la même qu'en algèbre conventionnelle. Faites par exemple déterminer par l'ordinateur le résultat de 1+2+3+4+5+6+7+8+9.

```
Ready
Print 1+2+3+4+5+6+7+8+9
 45
Ready
■
```

Le calcul entre parenthèses



Vous pouvez donc parfaitement faire exécuter des chaînes de calculs. Prenons un exemple pratique. Calculons le prix de revient total de trois tapis. Le prix du mètre carré est de 23 F 80. Le premier tapis mesure 2 m 45 sur 2 m 80, le deuxième 4 m 50 sur 3 m 85 et le troisième 2 m 75 sur 4 m 80. Il faut encore ajouter la TVA de 14%. Quel sera le prix de revient total de ces trois tapis ? Si vous placez les parenthèses là où il convient, une seule instruction PRINT suffit pour effectuer ce calcul :

```
print (2.45*2.8+4.5*3.85+2.75*4.8)*23.8*1.14
```

Cela a l'air beaucoup plus compliqué que ça n'est en réalité.

L'ordinateur calculera d'abord entre parenthèses la surface globale des trois tapis puis cette surface en mètres carrés sera multipliée par le prix au mètre carré. Le résultat obtenu sera alors multiplié par 1.14 pour ajouter la TVA. Tout ça avec une seule instruction. Si vous avez tout tapé correctement, vous obtenez 1041.33.

Comme vous avez déjà remarqué, c'est le point et non la virgule qui sert à séparer la partie entière d'un nombre de sa partie décimale. Si vous entrez par exemple 'print (9+23,8)*2', vous obtenez le message d'erreur Syntax error.

```
Ready
Print (9+23,8)*2
Syntax error
Ready
■
```

Ecriture exponentielle

Le CPC 6128 Plus ne peut calculer précisément que des nombres ne possédant pas plus de 9 chiffres. Peu importe d'ailleurs combien de ces chiffres sont avant ou après la virgule. Si vous obtenez un nombre décimal avec trois chiffres avant la virgule, l'ordinateur pourra calculer avec précision jusqu'à 6 décimales. Pour un nombre entier, vous ne pouvez donc pas faire représenter normalement de nombre supérieur à 999999999. Tout nombre supérieur sera présenté en écriture exponentielle. Si vous entrez par exemple 999999999+1, vous obtenez un nombre à 10 chiffres qui ne peut plus être représenté normalement.

```
Ready
Print 999999999+1
 1E+09
Ready
■
```

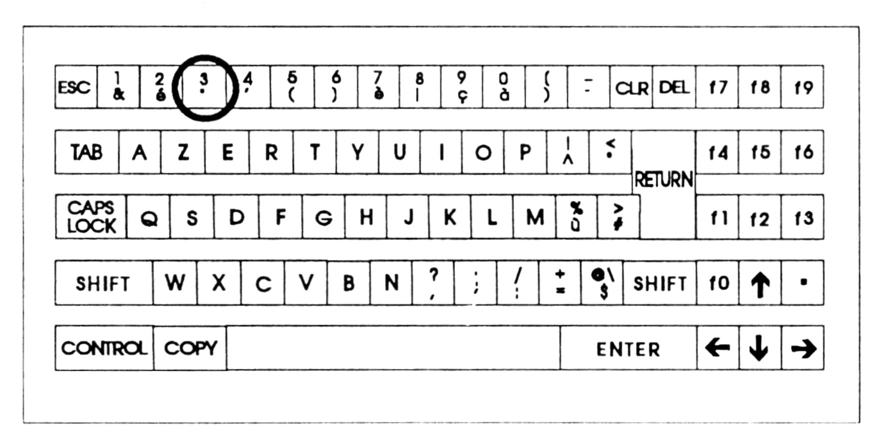
Ne soyez pas effrayé du résultat que vous obtenez. Cela signifie que le résultat est 1 multiplié par 10 puissance 9, c'est-à-dire un 1 plus 9 zéros. 'E+09' signifie base 10, exposant 9.

L'exposant peut être également négatif ainsi que le confirme l'exemple du calcul 'print 9/30000000000'.

```
Ready
Print 9/30000000000
 3E-10
Ready
■
```

Le résultat en écriture normale devrait être 0.0000000003 mais l'ordinateur utilise l'écriture exponentielle. '3E-10' signifie '3 multiplié par 10 puissance -10'. Le chiffre '3' se trouve donc placé en dixième position après la virgule. Nous en resterons là pour le moment en ce qui concerne les calculs en mode direct sur le CPC 6128 Plus. Vous trouverez une description des autres opérateurs mathématiques dans le manuel de votre ordinateur.

Sortie de texte avec Print



L'instruction PRINT vous permet non seulement de sortir des nombres mais également des textes, ce qu'on appelle des chaînes de caractères (string en anglais). Si vous vous précipitez pour faire de premières tentatives telles que 'print hallo' avant de lire nos explications, vous échouerez lamentablement.

```
Ready
Print hallo
 0
Ready
■
```

Bien que nous n'ayons pas obtenu le résultat escompté, l'ordinateur ne nous a pas envoyé de message d'erreur. Cela veut donc dire que l'ordinateur a bien exécuté cette instruction. Qu'a-t-il donc fait ? D'où vient le zéro ? Il est difficile

d'expliquer cela sans mettre la charrue avant les boeufs, sans aborder des thèmes que nous n'expliquerons en détail que plus tard. Disons simplement pour le moment que tout texte placé directement à la suite d'une instruction est considéré par l'ordinateur comme une variable. L'ordinateur a donc ici sorti le contenu de la variable qui s'appelle allo, donc 0 puisque nous n'avons encore affecté aucune valeur à la variable allo.

Il faut donc procéder autrement si nous voulons sortir un texte ou une chaîne de caractères avec l'instruction PRINT. Le texte ou la chaîne de caractères doit absolument être placé entre guillemets, comme ci-dessous :

```
print "allo"
```

Voilà ! Après une première tentative malheureuse, nous y sommes quand même arrivés. C'est peut-être le moment d'attirer votre attention sur le fait qu'il est plutôt nuisible d'être trop timide avec l'ordinateur. Quelles que soient les différences considérables entre les individus, il est en effet absolument certain que personne n'apprend la programmation autrement qu'en multipliant les expériences. Peu importe en réalité qu'une expérience soit réussie ou non, l'essentiel est vraiment d'accumuler les expériences. N'ayez donc surtout pas peur qu'une manoeuvre maladroite provoque des dégâts à votre ordinateur, c'est absolument impossible. Le pire qui puisse vous arriver, c'est une panne de courant qui vous ferait perdre instantanément toutes les données que vous avez entrées directement dans votre ordinateur. Nous vous expliquerons donc plus loin quelles précautions il faut suivre pour minimiser le risque de pertes de données.

Entrée simplifiée de Print

Vous pouvez remplacer l'instruction PRINT par un point d'interrogation, ce qui vous permet d'économiser pas mal de travail, étant donné que l'instruction PRINT est certainement la plus utilisée. '? 3*7' revient donc au même que 'PRINT 3*7'. De plus, il n'est pas nécessaire de faire suivre le point d'interrogation d'un espace. Essayez vous-même :

```
Ready
Print 3*7
 21
Ready
?3*7
21
Ready
■
```

A vous de voir si vous préférez employer cette abréviation. Lorsque vous utilisez votre ordinateur comme une calculatrice, cette abréviation vous permet par exemple d'effectuer des calculs aussi rapidement et facilement que sur une calculatrice de poche.

Pi et l'élévation à la puissance

Revenons aux mathématiques. La constante pi est certainement connue de tous, elle sert essentiellement à calculer les dimensions des cercles ou des sphères. Combien vaut pi ? 3.1412 ou 3.1214 ? Le plus souvent on arrondit à 3.14. Votre CPC 6128 Plus vous évite de chercher cette valeur. Essayez par exemple de faire sortir la valeur de pi sur l'écran. Entrez 'print pi'.

```
Ready
Print pi
 3.14159265
Ready
■
```

Vous voyez que votre ordinateur connaît la valeur de pi à 8 décimales près ! Vous pouvez donc calculer par exemple le périmètre d'un cercle de 12 cm de diamètre. La formule est Périmètre=Diamètre*pi.

```
Ready
Print 12*pi
 37.6991119
```

```
Ready
■
```

Le périmètre du cercle est donc environ 37.7 cm.

Voyons maintenant l'élévation à la puissance, c'est-à-dire la multiplication d'un nombre par lui-même. L'exposant détermine combien de fois la base doit être multipliée par elle-même.

2 puissance 3 signifie 2*2*2 donc 8
10 puissance 4 signifie 10*10*10*10 donc 10000

Dans le premier exemple, 2 est la base et 3 est l'exposant. L'élévation à la puissance s'écrit sur le CPC6 128 Plus en séparant la base de l'exposant par un accent circonflexe quise trouve à côté de la lettre P. Calculez par exemple 2 puissance 8.

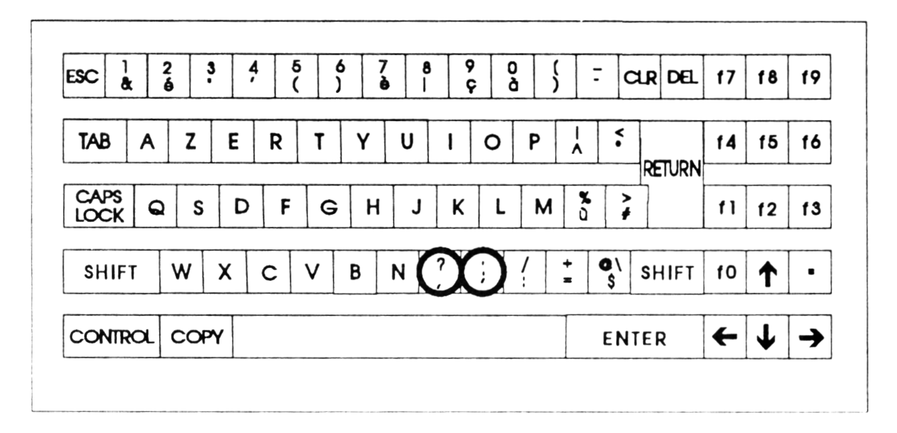
```
Ready
Print 2^8
 256
Ready
■
```

Ce n'est pas très difficile. Calculez maintenant la surface du cercle de 12 cm de diamètre. La formule est : surface=rayon au carré (rayon puissance 2) multiplié par pi (le rayon est la moitié du diamètre).

```
Ready
Print 6^2*pi
 113.097336
Ready
■
```

La surface du cercle est donc d'environ 113 centimètres carrés. Mais revenons à des choses plus intéressantes.

Combinaisons de chaînes de caractères et de nombres



Les touches ';' et ',' jouent un grand rôle dans la combinaison de chaînes de caractères (de textes) et de nombres. Si vous voulez par exemple faire sortir sur l'écran une ligne telle que "2.5 fois 2.5 = 6.25" en faisant en même temps calculer le résultat par l'ordinateur, il faut que l'instruction print sorte une chaîne de caractères et effectue ensuite un calcul. Voici comment cela peut être obtenu :

```
Ready
Print "2.5 fois 2.5 = ";2.5*2.5
2.5 fois 2.5 = 6.25
Ready
■
```

Le point-virgule permet donc de séparer une chaîne de caractères d'un nombre. Mais il permet également de séparer différentes chaînes de caractères ou différents calculs ou nombres les uns des autres. Essayez par exemple de faire calculer et sortir sur une seule ligne les puissances de 2 avec les exposants de 1 à 8.

```
Ready
Print 2^1;2^2;2^3;2^4;2^5;2^6;2^7;2^8
 2  4  8  16  32  64  128  256
```

```
Ready
■
```

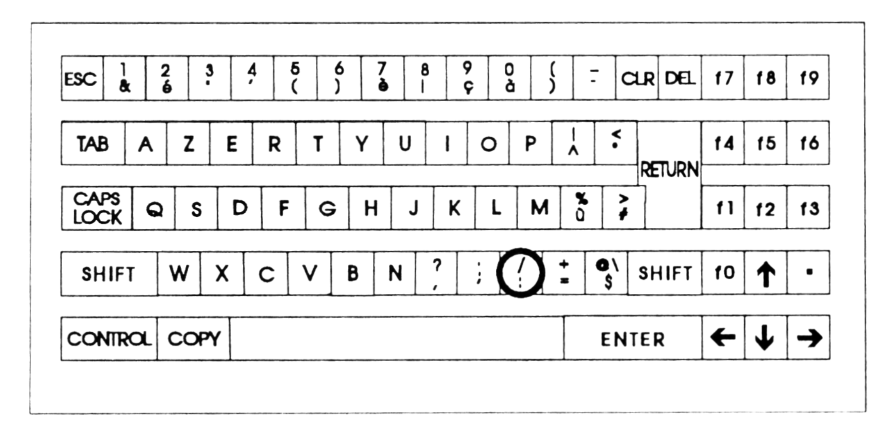
Le point-virgule placé dans une instruction print a donc pour effet que la prochaine expression (texte, calcul ou nombre) devant être sortie (par la même instruction print ou une autre instruction print) sera sortie immédiatement à la suite de la première expression et non pas sur la ligne suivante.

Entrez encore une fois la même instruction, mais en remplaçant cette fois les points-virgules par des virgules.

```
Ready
Print 2^1,2^2,2^3,2^4,2^5,2^6,2^7,2^8
 2          4          8
 16         32         64
 128        256
Ready
■
```

Vous voyez que la virgule permet également de séparer entre elles différentes expressions devant être sorties avec l'instruction print. Cependant, les expressions ne sont pas alors sorties immédiatement les unes à la suite des autres, mais en respectant un intervalle de 10 entre le début des différentes expressions. La virgule constitue donc un moyen simple pour réaliser des éditions formatées de colonnes de nombres. Elle est moins utilisée avec des chaînes de caractères.

Séparation des instructions



Une ligne d'instructions peut comporter au maximum 255 caractères. Le double-point permet de séparer entre elles différentes instructions placées sur une même ligne, aussi bien en mode direct qu'en mode programme. Voici pour plus de clarté un exemple, où vous allez effectuer en une seule fois deux calculs. Les résultats devront apparaître l'un en-dessous de l'autre. Nous calculerons 30^2*pi puis 30*pi. Essayez donc de faire exécuter ces deux calculs en entrant une seule ligne d'instruction que vous terminerez comme toujours par [Return].

```
Ready
Print 30^2*pi :Print 30*pi
 2827.43339
 94.2477796
Ready
■
```

Vous pouvez bien sûr entrer aussi 3 ou 4 instructions séparées entre elles par un double-point. Faites simplement attention à ce que votre ligne ne comprenne pas plus de 255 caractères.

L'exemple précédent va nous fournir maintenant un excellent moyen de constater l'effet du point-virgule. Videz l'écran et entrez à nouveau les deux instructions PRINT, en plaçant cette fois un point-virgule à la fin du premier PRINT.

```
Ready
Print 30^2*pi;:print 30*pi
 2827.43339  94.2477796
Ready
■
```

Le point-virgule après le premier PRINT empêche que l'ordinateur ne passe à la ligne suivante avant d'exécuter le second PRINT. Les espaces entre les nombres viennent du fait que l'ordinateur réserve automatiquement un caractère avant le premier chiffre de chaque nombre pour représenter le signe de ce nombre. Tout nombre positif est donc précédé d'un espace alors que les nombres négatifs sont précédés du signe -.

Notez enfin que le double-point ne doit pas nécessairement être suivi d'un espace mais que le nombre d'espaces suivant éventuellement le double-point est absolument sans effet sur l'instruction suivante.

3. Le premier programme

Nous allons essayer de vous faire faire maintenant vos premiers pas dans la programmation en Basic. N'ayez surtout pas peur de ne pas pouvoir suivre. Le langage Basic a été conçu pour pouvoir être utilisé par des débutants et il est inutile d'avoir trop de complexes par rapport à ce langage de programmation. Vous avez certainement constaté combien de jeunes enfants arrivent souvent facilement à se familiariser avec la programmation en Basic. L'essentiel est de se lancer un jour. Ce chapitre devrait donc être pour vous un petit pas dans le grand et passionnant monde de la programmation.

Qu'est-ce qu'un programme

En jargon de spécialiste, un programme est défini comme étant "une suite d'instructions pour résoudre un problème déterminé". Un programme n'est donc rien d'autre qu'une suite d'instructions placées dans l'ordre logique correct. La réalisation d'un programme passe donc par la définition d'un problème puis par la suite logique d'instructions pour aboutir au but, le programme. Un ordre logique correct d'instructions ne signifie pas uniquement, ce qui va de soi, que les instructions employées doivent être effectivement connues de l'ordinateur, mais également qu'elles doivent être placées dans un ordre tel que leur exécution dans cet ordre conduise effectivement à résoudre le problème posé.

La numérotation des lignes

Un programme est donc une suite d'instructions. Mais comment l'ordre de ces instructions est-il déterminé ? En Basic, chaque ligne d'instructions doit être dotée d'un numéro de ligne qui détermine dans quel ordre se déroulera le programme.

Représentez-vous par exemple le programme mental qui peut vous permettre de trouver avec votre calculatrice de poche combien font 48/12. Vous verrez que votre cerveau doit

également suivre un certain ordre dans l'exécution des différentes tâches qui vous permettront de résoudre ce problème :

1. rechercher la calculatrice
2. allumer la calculatrice
3. si piles à plat, aller au point 12
4. appuyer sur la touche '4'
5. appuyer sur la touche '8'
6. appuyer sur le signe de la division
7. appuyer sur la touche '1'
8. appuyer sur la touche '2'
9. appuyer sur le signe '='
10. lire le résultat
11. mémoriser le résultat
12. éteindre la calculatrice

Vous voyez qu'une tâche aussi simple peut en fait être subdivisée en de nombreuses étapes intermédiaires. Vous voyez que les différentes étapes ont été numérotées pour déterminer l'ordre dans lequel elles seront exécutées. Il y a cependant une autre raison importante à la numérotation, comme nous le montre le point 3. Le point 3 présente en effet un exemple d'une notion très importante en logique de programmation, l'instruction de saut. Si une condition déterminée est remplie (piles à plat ?), le programme saute directement au point 12. Le nombre 12 est ici ce qu'on appelle l'adresse d'une instruction. Sans numéros de lignes, il ne serait donc pas possible de sauter certaines instructions.

Nous avons déjà rencontré au chapitre précédent une instruction, l'instruction PRINT, qui permet de sortir des données sur l'écran. Mais nous avions utilisé cette instruction en mode direct, c'est-à-dire que l'instruction était exécutée directement après que nous avions appuyé sur la touche [Return]. Nous allons maintenant créer un programme à partir de trois instructions PRINT.

Comment allons-nous procéder ? Simplement en dotant chaque instruction d'un numéro de ligne qui déterminera dans quel ordre le programme devra se dérouler.

LES NUMEROS DE LIGNE DOIVENT ETRE COMPRIS SUR LE CPC 6128 Plus ENTRE 1 ET 65535 INCLUS. L'ESPACEMENT ENTRE LES NUMEROS DE LIGNE EST SANS IMPORTANCE.

Un programme peut donc se composer de lignes ayant les numéros 1, 8, 10, 20, 100, 200, 300, etc... La seule recommandation que nous pouvons vous faire à cet égard est d'éviter d'avoir des lignes de programme dont les numéros se suivent immédiatement (100, 101, 102, ...). En effet, si vous voulez ensuite insérer une ligne entre deux lignes existantes, il faut que ces lignes soient au moins espacées de 2. Par exemple, si vous écrivez une ligne portant le numéro 11 puis une ligne portant le numéro 12, vous ne pouvez insérer aucune ligne entre ces deux lignes. Si par contre vos deux lignes portent les numéros 11 et 13, vous pouvez ensuite écrire une ligne avec le numéro 12 et l'ordinateur intercalera automatiquement cette ligne 12 entre les lignes 11 et 13. Dans la pratique, il est recommandé d'espacer les numéros de ligne de 10 en 10.

Voici maintenant le problème que devra résoudre notre premier programme :

1. Vider l'écran
2. Sortir le texte "18 DIVISE PAR 6 DONNE:" sur l'écran
3. Sortir le résultat sur l'écran, immédiatement à la suite du texte

Ce problème devra être résolu en trois lignes de Basic. Comment attaquer ce problème ? Il faut d'abord choisir le numéro de ligne. Choisissons par exemple 100. Il nous suffit maintenant d'entrer le nombre 100 suivi de l'instruction pour vider l'écran. Souvenez-vous que cette instruction est CLS. Notre ligne de programme se présentera maintenant ainsi :

```
Ready
 100 cls
■
```

NE FAITES PAS SUIVRE LE NUMERO DE LIGNE D'UN ESPACE, CET ESPACE SERA DE TOUTE FACON AJOUTE PAR L'ORDINATEUR AUTOMATIQUEMENT.

C'était donc la première ligne de notre programme. La seconde doit sortir une chaîne de caractères. C'est l'instruction PRINT qu'il nous faut donc utiliser. Comme nous travaillons avec un espacement de 10, notre deuxième ligne recevra le numéro 110. Entrez maintenant cette ligne.

```
Ready
100 cls
110 print "18 divisé par 6 donne";
■
```

Nous avons placé un point-virgule à la suite de notre chaîne de caractères pour que l'instruction PRINT suivante soit exécutée sur la même ligne, immédiatement après le texte.

Essayez d'écrire vous-même la troisième ligne qui doit calculer 18/6 et sortir le résultat de ce calcul.

```
Ready
100 cls
110 print "18 divisé par 6 donne";
120 print 18/6
■
```

Le programme est maintenant terminé et il ne reste plus qu'à le faire exécuter.

Lancement d'un programme

Le CPC 6128 Plus n'a pas simplement placé les lignes que vous avez entrées sur l'écran, il les a également conservées dans sa mémoire Basic. même si vous videz maintenant l'écran, le programme ne sera pas perdu. Vous pourrez le faire exécuter à tout moment.

L'INSTRUCTION 'RUN' LANCE UN PROGRAMME BASIC

Entrez maintenant l'instruction 'RUN' et voyez ce qui se passe sur l'écran.

```
18 divisé par 6 donne 3
Ready
■
```

Ce programme est un premier succès pour un débutant en programmation.

Vous pouvez maintenant lancer ce programme aussi souvent que vous le voulez. Voyez vous-même : entrez encore une fois 'RUN' et encore une autre...

Modification du programme

Il peut arriver que vous ayez à modifier un programme. Supposons par exemple que vous vouliez maintenant calculer avec le programme que vous venez d'entrer, non plus 18/6 mais 24/6. Il n'est pas nécessaire pour cela d'entrer à nouveau les trois lignes de programme. Il vous suffit en effet de modifier les lignes existantes aux endroits nécessaires. Pour cela, il faut d'abord que vous fassiez apparaître le programme sur l'écran.

L'instruction 'List' montre les lignes du programme sur l'écran

Entrez maintenant LIST suivi bien sûr de [Return] et regardez ce qui se produit. Les lignes de programme que vous venez d'entrer apparaissent maintenant sur l'écran. Vous remarquez que votre ordinateur a écrit toutes les instructions en majuscules. Cela permet d'obtenir une plus grande clarté dans les listings.

Le "listing" est un programme sorti soit sur écran, soit sur imprimante.

Il faut encore que vous sachiez que lors du listing de programmes longs, l'écran réalise un "scrolling", c'est-à-dire qu'une ligne de programme est ajoutée au fur et à mesure en

bas de l'écran, que toutes les lignes remontent simultanément et que la ligne en haut de l'écran disparaît en même temps. Comme les lignes de programme défilent alors plus vite que vous ne pouvez les lire, vous pouvez interrompre le défilement à tout moment en appuyant sur la touche [ESC]. Si vous appuyez ensuite sur n'importe quelle autre touche, le défilement se poursuivra. Par contre, si vous appuyez deux fois de suite sur la touche [ESC], le défilement est interrompu définitivement. Il ne peut plus être relancé qu'en entrant à nouveau l'instruction LIST.

Revenons maintenant à la modification du programme. Pour modifier les lignes de programme pour calculer 24/6, vous disposez de deux possibilités :

1. La méthode avec le curseur de copie
2. La méthode avec l'instruction EDIT

Vous avez déjà été confronté au curseur de copie au début de cet ouvrage. Cela ne doit pas nous empêcher de rappeler une nouvelle fois la marche à suivre. La ligne à modifier se trouve sur l'écran. Utilisez les touches curseur en tenant la touche [Shift] enfoncée pour déplacer le curseur de copie vers le début de la ligne à modifier (ligne 110) :

```
Ready
list
100 cls
■10 print "18 divisé par 6 donne";
120 print 18/6
Ready
■
```

Appuyez maintenant sur la touche COPY jusqu'à ce que le curseur atteigne le premier caractère à modifier :

```
Ready
list
100 cls
110 print "■8 divisé par 6 donne";
120 print 18/6
```

```
Ready
110 print "■
```

C'est ici qu'intervient la première modification de la ligne. Entrez maintenant le nombre 24. Déplacez maintenant le curseur de copie vers le premier espace suivant le 18 (n'oubliez pas la touche [Shift] !).

```
Ready
list
100 CLS
110 PRINT "■8 divisé par 6 donne";
120 PRINT 18/6
Ready
110 print "24■
```

Vous pouvez maintenant copier le reste de la ligne avec la touche COPY puis appuyer sur [Return]. La ligne est maintenant modifiée. Mais pour que le résultat sorti corresponde effectivement au texte, il nous faut encore modifier la ligne 120. Procédez comme pour la ligne 110. Il n'est certes pas très aisé d'arriver à maîtriser le curseur de copie, mais c'est simplement une question d'habitude et d'entraînement.

Passons maintenant à la seconde méthode de modification d'un programme. L'instruction EDIT appelle une ligne déterminée à l'écran pour permettre sa modification. Entrez maintenant l'instruction EDIT 110 (n'oubliez pas [Return] !) :

```
Ready
edit 110
110 ■RINT "24 divisé par 6 donne";
```

La ligne 110 est maintenant affichée et le curseur figure sur la lettre P. Amenez maintenant le curseur sur le premier caractère à modifier. Essayons par exemple de transformer à nouveau le 24 en 18. Si vous entrez maintenant 18, vous obtenez :

```
Ready
edit 110
110 print "18■4 divisé par 6 donne";
```

Le nombre 24 n'a pas été effacé par le nombre 18. Le nombre 18 a été inséré devant le nombre 24. C'est que l'instruction EDIT travaille en effet systématiquement en mode d'insertion (sauf si l'on appuie simultanément sur les touches [Control] et [Tab], ce qui a pour effet de commuter en mode sur-impression). Il nous faut donc encore supprimer le nombre 24. Rappelez-vous que c'est la touche CLR qui permet de supprimer le caractère placé sous le curseur. Appuyez deux fois sur cette touche :

```
Ready
edit 110
110 print "18█divisé par 6 donne";
```

Une fois que vous avez effectué toutes les modifications nécessaires sur la ligne, appuyez sur la touche [Return].

Vous pouvez vérifier que la ligne a bien été modifiée en entrant maintenant l'instruction LIST.

Nous vous proposons ainsi deux méthodes pour modifier un programme. A vous de choisir celle que vous préférez employer.

Branchements

Il est rare que les programmes soient exécutés systématiquement de haut en bas comme dans l'exemple précédent. Il est au contraire fréquent que le programme saute certaines lignes, le plus souvent sous certaines conditions, pour poursuivre l'exécution du programme dans un autre endroit. Dans ce cas, l'instruction de saut est nécessaire.

L'INSTRUCTION 'GOTO' SAUTE AU NUMERO DE LIGNE INDIQUE

En Basic, c'est l'instruction GOTO qui permet de sauter à n'importe quel numéro de ligne.

Voyons tout de suite un exemple concret. Comme vous le savez, notre programme d'exemple se termine par la ligne 120.

Que se passera-t-il si nous sautons ensuite, par exemple en ligne 130, à la ligne 110 ? Je prétends que le programme sortira indéfiniment "18 DIVISE PAR 6 DONNE 3". Mais voyez vous-même. Entrez la ligne '130 GOTO 110' et lancez le programme avec 'RUN'.

```
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
18 divisé par 6 donne 3
```

Nous avons créé avec la ligne 130 ce qu'on appelle une boucle sans fin car le programme parcourt indéfiniment les lignes 110 à 130. Si nous n'intervenons pas, le programme continuera ainsi à tourner jusqu'à la saint-glinglin.

LA TOUCHE ESC INTERROMPT LE DEROULEMENT DU PROGRAMME

Si vous appuyez maintenant sur la touche [ESC], le programme s'interrompt. Si vous appuyez ensuite sur n'importe quelle autre touche, l'exécution du programme reprend. Si vous voulez arrêter le programme définitivement, vous devez appuyer deux fois sur la touche [ESC]. L'ordinateur affiche alors 'Break in ...' et indique ainsi en quelle ligne le programme a été interrompu. Dans l'exemple qui nous intéresse, la ligne à

laquelle le programme sera effectivement arrêté dépend uniquement du hasard.

Si, après avoir interrompu définitivement le déroulement du programme, vous voulez qu'il reprenne malgré tout, il faut utiliser l'instruction 'CONT' (=continuer).

L'INSTRUCTION 'CONT' POURSUIT L'EXECUTION DU PROGRAMME APRES UNE INTERRUPTION

Entrez donc cette instruction. Le programme retombe à nouveau dans la boucle sans fin et vous ne pouvez l'arrêter qu'avec la touche [ESC].

Sauvegarde et chargement de programmes

Si vous éteignez maintenant votre ordinateur, le programme sera perdu. Comme il est pénible de devoir taper à nouveau un programme chaque fois qu'on veut s'en servir, les programmes peuvent être sauvegardés sur disquette grâce au lecteur de disquette intégré. Sauvegardons donc notre programme. Introduisez tout d'abord une disquette formatée dans le lecteur de disquette (le formatage des disquettes est expliqué au chapitre 8). Choisissez maintenant un nom pour votre programme. Ce nom doit comprendre au maximum 8 caractères. Prenons par exemple le nom "TEST". L'instruction de sauvegarde d'un programme est 'SAVE'. Entrez donc :

```
Ready
save "test"■
```

Pour recharger plus tard ce programme et le lancer, vous devez utiliser l'instruction LOAD.

```
Ready
load "test"■
```

Si vous avez oublié le nom du programme, vous pouvez faire afficher à l'écran le contenu de la disquette, avec l'instruction CAT.

Il vous faut ensuite entrer l'instruction RUN pour lancer le programme. Remarquez également que vous pouvez ajouter à l'instruction RUN un numéro de ligne. Le programme ne sera pas alors lancé comme habituellement à partir de la première ligne, mais à partir de la ligne dont le numéro a été placé à la suite de RUN.

Nous n'en dirons pas plus pour l'instant et nous reviendrons plus en détail sur les possibilités du lecteur de disquette dans le chapitre que nous lui avons consacré.

Suppression d'un programme

Comme nous l'avons déjà indiqué, le programme est supprimé lorsque vous éteignez l'ordinateur. Il existe cependant également une instruction qui vous permet de supprimer un programme sans couper l'ordinateur.

L'INSTRUCTION 'NEW' SUPPRIME LE PROGRAMME SE TROUVANT EN MEMOIRE

Si vous entrez maintenant 'NEW', le programme sera supprimé. Il va de soi qu'il faut sauvegarder un programme avant d'utiliser cette instruction si vous ne voulez pas être obligé de le retaper plus tard.

Pour nos lecteurs sceptiques : entrez l'instruction LIST après l'instruction NEW et vous verrez que le programme a vraiment été supprimé puisque le listing ne vous montre aucune ligne. L'instruction RUN est également sans effet.

Vous avez acquis maintenant les premières connaissances de base nécessaires pour réaliser un programme. C'est sur ces bases que reposera le travail des chapitres suivants. Si donc vous vous êtes senti dépassé par certaines notions développées dans ce chapitre, nous vous recommandons de l'étudier à nouveau attentivement.

4. Aides à la programmation

C'est la puissance du Basic de votre CPC 6128 Plus dont vous pouvez vraiment être fier qui justifie le présent chapitre. Peu d'ordinateurs présentent dans la version de base des instructions facilitant à ce point la création et le traitement de programmes. Ce chapitre est donc essentiel pour que vous utilisiez à fond les facilités que vous offre votre ordinateur pour travailler agréablement au développement de vos programmes.

Numérotation automatique des lignes

Nous vous avons conseillé de numéroter les lignes de vos programmes de 10 en 10. Mais vous constaterez vite si vous en faites l'expérience qu'il est pénible de devoir penser en début de chaque ligne à rajouter un numéro de ligne. C'est pourquoi les concepteurs du Basic de votre ordinateur ont développé une instruction très pratique de numérotation automatique des lignes. Voici d'abord une description de cette instruction :

Problème :	Numérotation automatique des lignes	
Instruction :	AUTO début,espacement	
Paramètres :	début espacement	numéro de la première ligne écart entre deux numéros
Exemple :	AUTO 100,10 Toutes les lignes seront numérotées de 10 en 10 et la première ligne sera la ligne 100 (100, 110, 120, 130, 140...)	
Remarque :	La numérotation automatique des lignes peut être interrompue avec [ESC].	

Nous avons donné ici pour la première fois une description détaillée d'une instruction comme nous le ferons désormais pour toutes les instructions que nous serons maintenant amenés

à rencontrer. Peut-être ne comprenez-vous pas très bien ce que signifie le mot paramètre. Un paramètre est une expression quelconque qu'on "attelle" à la suite d'une instruction. L'instruction LOAD exige par exemple un paramètre puisqu'elle doit être suivie d'un nom de programme.

Mais revenons maintenant à l'instruction AUTO. Supposons que vous vouliez commencer à entrer des lignes de programme en commençant par la ligne 10 et que vous vouliez que les numéros de ligne se suivent de 5 en 5. Vous entrerez donc l'instruction :

```
AUTO 10,5
```

Vous voyez alors apparaître immédiatement le numéro de la première ligne et l'ordinateur attend maintenant que vous entriez cette ligne. Entrez par exemple 'PRINT "Ligne 10"' et terminez la ligne avec [Return]. L'ordinateur sort alors le numéro de la seconde ligne (15) et attend de nouveau que vous entriez cette ligne. Vous pouvez mettre fin à la numérotation automatique des lignes avec la touche [ESC]. Le message Ready vous indique alors que la numérotation automatique n'est plus activée. Entrez alors 'AUTO 10,5' pour constater une des particularités de cette instruction. L'ordinateur réagit ainsi :

```
10 PRINT "Ligne 10"
```

Vous voyez donc, que le CPC 6128 Plus est très serviable puisque, s'il existe déjà une ligne portant le même numéro que celui

obtenu par la numérotation automatique, il vous présente le texte de la ligne déjà entrée. Ceci évite que vous ne modifiez des lignes par erreur. Si vous voulez que la ligne soit conservée dans son ancienne version, appuyez simplement sur [Return]. Mais si vous voulez modifier cette ligne, entrez simplement le texte de la nouvelle ligne puis appuyez sur [Return]. Essayez cette possibilité : conservez inchangée la ligne 10 mais, arrivé à la ligne 15, entrez 'PRINT "Nouvelle ligne 15"'. Interrompez ensuite la numérotation automatique avec la touche [ESC] puis examinez le résultat en entrant l'instruction LIST.

Vous disposez ainsi d'une possibilité vous permettant de vous simplifier l'entrée de programmes dans votre ordinateur. Nous vous invitons donc à utiliser cette possibilité également pour entrer les programmes que nous vous fournissons en exemple dans cet ouvrage.

Renumérotation des lignes

L'espacement entre les numéros de ligne n'a aucune influence sur le déroulement d'un programme. À vous de décider si vous préférez numéroter vos programmes avec un espacement de 1, de 5 ou de 10. Il faut simplement tenir compte du fait qu'un écart trop faible entre les numéros de ligne d'un programme rend plus difficile l'insertion de nouvelles lignes dans un programme. Entrez l'exemple suivant pour mieux comprendre le problème :

```
1 CLS
2 PRINT "Multiplions deux nombres:"
3 PRINT "Le premier nombre sera 15"
4 PRINT "Le deuxieme nombre sera 12"
5 PRINT "Le resultat est";15*12
```

Comme vous voyez, nous avons numéroté ce programme avec un écart de 1. Le programme produit le résultat suivant :

```
Multiplions deux nombres:
Le premier nombre sera 15
Le deuxieme nombre sera 12
Le resultat est 180
```

Pour améliorer la présentation, nous allons maintenant souligner la première ligne. L'image obtenue à l'écran devra donc être maintenant la suivante :

```
Multiplions deux nombres:
--------------------------
Le premier nombre sera 15
Le deuxieme nombre sera 12
Le resultat est 180
```

Pour obtenir ce résultat, il nous faut ajouter une ligne de programme supplémentaire à la suite de la ligne 2. Il est donc nécessaire d'augmenter de 1 les numéros des lignes placées après la ligne 2 pour que nous puissions donner à la nouvelle ligne le numéro 3. Le programme complet sera alors :

```
1 CLS
2 PRINT "Multiplions deux nombres:"
3 PRINT "-------------------------"
4 PRINT "Le premier nombre sera 15"
5 PRINT "Le deuxieme nombre sera 12"
6 PRINT "Le resultat est";15*12
```

Bien sûr vous pourriez modifier le numéro des 3 lignes 3 à 5 en 4 à 6 en utilisant pour cela une des méthodes de modification des lignes de programme que nous vous avons indiquées. Il existe heureusement une méthode encore plus simple grâce à une instruction très pratique dont dispose le puissant Basic de votre ordinateur. Voyons tout d'abord la description de cette instruction :

Problème :	Renumérotation des lignes	
Instruction :	RENUM nouvelle,ancienne,espacement	
Paramètres :	nouvelle	nouveau numéro de la première ligne à renuméroter.
	ancienne	ancien numéro de la première ligne à numéroter.
	espacement	nouvel écart entre deux lignes renumérotées
Exemple :	RENUM 100,10,5 Toutes les lignes seront renumérotées de 5 en 5 à partir de la ligne 10 du programme qui portera maintenant le numéro 100	

Remarque : La renumérotation ne doit pas avoir pour effet que le dernier numéro de ligne dépasse 65535, sinon l'instruction RENUM ne sera pas exécutée et le message d'erreur "Improper argument" apparaîtra !

Ne vous découragez pas si cette instruction vous semble un peu plus complexe que celles que nous avons vues jusqu'ici. Vous voyez que vous pouvez faire suivre cette instruction de trois paramètres : le numéro qui doit être attribué à la première ligne à renuméroter, le numéro de la première ligne à partir de laquelle le programme doit être renuméroté (vous n'êtes pas obligé de renuméroter le programme tout entier) et le nouvel écart entre les différents numéros de ligne.

Quels paramètres devrons-nous utiliser avec l'instruction RENUM pour transformer les numéros des lignes 3 à 5 en 4 à 6 ? Procédons méthodiquement. Il nous faut indiquer tout d'abord le nouveau numéro de la première ligne à renuméroter. Ce numéro sera 4. L'ancien numéro de la première ligne à renuméroter est 3 et l'écart restera de 1. Nous obtenons donc l'instruction suivante :

```
RENUM 4,3,1
```

Cette instruction nous conduit au succès que vous pouvez constater en entrant l'instruction LIST :

```
1 CLS
2 PRINT "Multiplions deux nombres:"
4 PRINT "Le premier nombre sera 15"
5 PRINT "Le deuxieme nombre sera 12"
6 PRINT "Le resultat est";15*12
```

Nous avons, avec une seule instruction, transformé le programme, de sorte que nous avons maintenant la place nécessaire pour entrer une nouvelle ligne 3. Vous pouvez donc entrer :

```
3 PRINT "-----"
```

Vous pouvez vous éviter à l'avenir de telles manipulations en travaillant systématiquement avec des numéros de ligne espacés de 10 en 10. Essayez maintenant de trouver quelle forme devra prendre l'instruction RENUM pour que notre programme d'exemple soit numéroté de 10 en 10 et commence par la ligne 100 ? La solution est :

```
RENUM 100,1,10
```

L'instruction RENUM ne se contente pas toutefois de renuméroter les lignes. Elle accomplit en outre une tâche très importante puisqu'elle modifie les instructions de saut (comme GOTO) de façon à tenir compte des nouveaux numéros de ligne. Voyons ce que cela signifie concrètement. Ajoutez à votre programme une ligne supplémentaire comme dans l'exemple ci-dessous :

```
100 CLS
110 PRINT "Multiplions deux nombres:"
120 PRINT "-------------------------"
130 PRINT "Le premier nombre sera 15"
140 PRINT "Le deuxieme nombre sera 12"
150 PRINT "Le resultat est";15*12
160 GOTO 110
```

Notre dernière instruction crée une boucle sans fin. Si nous renumérotons le programme, la ligne 110 recevra un autre numéro. Il ne faudra donc plus que l'instruction GOTO saute à la ligne 110 car cette ligne n'existera plus après la renumérotation. C'est pourquoi l'instruction RENUM ne modifie pas seulement les numéros de ligne au début de chaque ligne, mais également dans chaque instruction de saut à une ligne dont le numéro a été modifié. C'est très important car sinon l'instruction RENUM serait inutilisable. Vous pouvez vous convaincre vous-même du fonctionnement de cette instruction en entrant l'instruction 'RENUM 10,100,5'. Si vous entrez ensuite l'instruction LIST, vous constaterez que l'adresse de saut à la suite de l'instruction GOTO a été modifiée de façon à correspondre au nouveau numéro de la ligne 160. RENUM peut être utilisée seule sans paramètres. Dans ce cas, le

programme sera entièrement renuméroté à partir de la ligne 10 avec un pas de 10.

Cette instruction vous sera certainement très utile à l'avenir.

Suppression sélective de lignes

Vous connaissez déjà l'instruction NEW qui supprime le programme se trouvant en mémoire. Mais comment faire si vous ne voulez pas supprimer la totalité du programme mais seulement une ou plusieurs lignes de ce programme ? La suppression d'une ligne est très facile : il suffit d'entrer le numéro de cette ligne puis d'appuyer sur [Return]. Pour tester cette possibilité de supprimer une ligne, supprimez le programme en mémoire, et entrez le programme suivant en vous servant de l'instruction AUTO :

```
10 PRINT
20 PRINT
30 PRINT
40 PRINT
50 PRINT
60 PRINT
70 PRINT
80 PRINT
90 PRINT
```

Ce programme ne sort que 9 lignes vides sur l'écran, mais cela suffit pour les exercices suivants. Supprimez maintenant la ligne 50 en entrant 50 suivi de [Return]. Si vous listez maintenant le programme, vous voyez que la ligne 50 n'existe plus. Mais si vous avez plusieurs lignes à supprimer, cette méthode peut se révéler longue et ennuyeuse. Une instruction vous facilite le travail :

Problème :	Suppression sélective de lignes
Instruction :	DELETE début,fin

Paramètres :	début fin	première ligne à supprimer dernière ligne à supprimer
Exemple :	DELETE 40-130 supprimer les lignes 40 à 130	

Pour supprimer les lignes 20 à 40 de notre programme, il faut donc entrer l'instruction :

```
DELETE 20-40
```

Il y a plusieurs possibilités de combiner les deux paramètres que peut recevoir cette instruction. Le tableau suivant illustre ces diverses possibilités :

DELETE 10	supprime la ligne 10
DELETE 100-150	supprime les lignes 100 à 150
DELETE 150-	supprime toutes les lignes à partir de 150 inclus
DELETE -100	supprime toutes les lignes jusqu'à 100 inclus

Ces différentes possibilités de délimiter la section du programme à laquelle doit s'appliquer l'instruction DELETE existent également pour l'instruction LIST ainsi que le montre le tableau suivant :

LIST 10	liste la ligne 10
LIST 100-150	liste les lignes 100 à 150
LIST 150-	liste toutes les lignes à partir de 150 inclus
LIST -100	liste toutes les lignes jusqu'à 100 inclus

Les touches de fonction

Nous allons maintenant faire connaissance avec une des aides à la programmation les plus pratiques de votre CPC 6128 Plus. Vous constaterez vite que lorsqu'on programme, il arrive souvent qu'on ait à entrer particulièrement fréquemment certaines instructions. Votre ordinateur vous permet d'entrer certaines instructions ou certains textes en appuyant sur une

seule touche. Vous pouvez utiliser à cet effet les touches du pavé numérique prévu pour cela.

Vous pouvez en effet affecter à chacune de ces touches une suite d'au maximum 32 caractères sans dépasser toutefois un nombre total de 120 caractères pour toutes les touches ainsi définies. Voici la description de l'instruction qu'il faut utiliser pour cela :

Problème :	Affectation des touches de fonction	
Instruction :	KEY n,chaîne	
Paramètres :	n chaîne	numéro de la touche de fonction chaîne de caractères (max. 32 caractères)
Exemple :	KEY 0,"LIST" Affecte l'instruction LIST à la touche f0.	
Remarque :	Le nombre global des caractères ainsi affectés aux touches de fonction ne doit pas dépasser 120	

Cette instruction est très intéressante, non ? Essayons tout de suite de l'utiliser. Il y a 13 touches de fonction numérotées de 0 à 12. La touche [Enter] peut recevoir deux fonctions, l'une lorsqu'elle est actionnée normalement et l'autre lorsqu'elle est actionnée en même temps que la touche [Control]. L'illustration ci-dessous vous indique les numéros des différentes touches de fonction :

	7	8	9
RETURN	4	5	6
	1	2	3
SHIFT	0	↑	10
11 / 12	←	↓	→

La touche '.' porte donc le numéro 10, la touche [Enter] le numéro 11 et la combinaison [Control] + [Enter] le numéro 12. Essayons maintenant d'affecter des instructions intéressantes à quelques touches de fonction. L'instruction LIST est utilisée très souvent. Affectons-la à la touche de fonction 0 avec l'instruction suivante :

```
KEY 0,"LIST"
```

N'oubliez pas l'espace qui doit séparer l'instruction des paramètres, sinon votre ordinateur protestera (Syntax error). Vous pouvez maintenant appuyer sur la touche 0 et vous voyez apparaître l'instruction LIST sur l'écran. Cette instruction n'est pourtant pas exécutée pour le moment. Il faut en effet que vous appuyiez maintenant sur la touche [Return]. Mais si vous voulez que l'instruction LIST soit exécutée directement chaque fois que vous appuyez sur la touche de fonction 0, sans que

vous soyez obligé d'appuyer en plus sur [Return], il faut que vous ajoutiez le code de la touche [Return] à la définition de la touche 0. Il faut utiliser pour cela une instruction que nous étudierons plus tard. Le code de la touche [Return] est 13. L'instruction se présente maintenant ainsi :

```
KEY 0,"LIST"+CHR$(13)
```

Ne vous laissez pas effrayer par les nouveautés qui apparaissent dans cette instruction. Le principal pour le moment est que vous constatiez l'effet qu'elle produit : si vous appuyez maintenant sur la touche 0, non seulement l'instruction LIST apparaît sur l'écran mais elle est en plus exécutée immédiatement.

Il y a cependant des mots-instructions auxquels il vaut mieux ne pas ajouter [Return]. Par exemple NEW. Si vous affectiez en effet l'instruction NEW plus [Return] à une touche de fonction, il suffirait que vous appuyiez par mégarde sur cette touche pour que tout le programme en mémoire soit supprimé. Faites donc attention à n'ajouter [Return] qu'à des instructions dont l'utilisation par erreur soit sans conséquence.

Nous avons ainsi appris le code de [Return]. Un autre code important par rapport aux touches de fonction est celui des guillemets. Essayez par exemple d'affecter l'instruction 'RUN"' que nous avons étudiée plus haut à la touche de fonction 1 ! Où est le problème ? L'instruction correcte devrait être 'KEY 1,"RUN""' et pourtant, si vous l'entrez, vous obtenez un Syntax error. En effet l'ordinateur considère avec sa logique implacable et un peu simple que les seconds guillemets ne peuvent être que la fin de la chaîne de caractères. Il ne sait donc pas du tout ce que peuvent signifier les troisièmes guillemets. Pour nous tirer d'affaire, nous devons donc utiliser le code des guillemets. L'instruction correcte sera donc :

```
KEY 1,"RUN"+CHR$(34)
```

Si nous voulons ajouter encore le nom d'un programme que nous utilisons souvent plus [Return], nous obtenons une instruction vraiment compliquée :

```
KEY 1,"RUN"+CHR$(34)+"TEST"+CHR$(34)+CHR$(13)
```

Essayez simplement de retenir pour le moment à quoi correspondent les deux codes 13 et 34 !

Vous choisirez avec le temps une affectation standard des touches de fonction que vous n'aurez plus besoin d'entrer chaque fois que vous allumerez votre ordinateur. Vous pouvez en effet placer une série d'instructions KEY dans un programme que vous sauvegarderez sur disquette pour pouvoir le charger en premier chaque fois que vous commencerez à travailler sur votre ordinateur. Voici comment pourrait se présenter un tel programme :

```
10 KEY 0,"LIST"+CHR$(13)
20 KEY 1,"PRINT "
30 KEY 2,"NEW"
40 KEY 3,"RENUM "
50 KEY 4,"AUTO "
60 KEY 5,"DELETE "
```

Essayez de mettre au point votre propre programme de touches de fonction pendant ou après l'étude du chapitre suivant !

Encore une indication importante pour terminer : le coup de trois touches ([Shift]/[Control]/[ESC]) supprime toutes les touches de fonction !

5. Introduction au Basic

Ce chapitre est une introduction au langage de programmation Basic. Ce n'est donc pas une simple présentation des instructions Basic car nous découvrirons les instructions Basic en essayant en même temps de réaliser pas à pas un programme de gestion d'adresses qui nous permettra d'illustrer par des exemples le fonctionnement des diverses instructions. Ce chapitre ne fera pas de vous des programmeurs chevronnés mais il vous permettra de pénétrer sérieusement dans le domaine de la programmation appliquée.

DESCRIPTION DU PROBLEME POUR UNE GESTION D'ADRESSES

Les programmes de gestion d'adresses sont une application favorite des possibilités des ordinateurs familiaux car tout le monde a besoin de gérer sous une forme ou sous une autre un certain nombre d'adresses. Bien sûr, si vous n'avez que quelques adresses à gérer, il est probable que le système le plus pratique et le plus rapide est le carnet d'adresses traditionnel. Par contre, dès que vous avez un nombre plus important d'adresses à enregistrer, il devient intéressant d'utiliser un ordinateur et de toute façon une gestion d'adresses sur ordinateur est tout de même plus impressionnante.

Quelles sont les possibilités qu'on peut attendre d'un tel programme ? En tout état de cause, les adresses doivent pouvoir être entrées et sorties.

Avant d'exiger d'autres fonctions du programme, il nous faut encore préciser un point. Dans une gestion de données, on distingue toujours deux parties essentielles :

1. Le programme
2. Les données

Les données ne font partie à proprement parler du programme. En effet seul le programmeur (et non pas l'utilisateur) peut

placer des données dans un programme. Une modification des données ne pourrait donc se faire qu'en modifiant le programme, ce qui n'est bien sûr pas acceptable.

Le programme se trouve donc sur un périphérique de stockage (disquette) et il est chargé dans l'ordinateur lorsqu'on en a besoin. Mais où placer les données ?

Il nous faut organiser un fichier. Un fichier est une collection de données placée sur un périphérique de stockage. Il y a plusieurs modes d'organisation des données, mais nous ne nous intéresserons qu'au plus simple, le fichier séquentiel. Séquentiel signifie que les données sont rangées les unes à la suite des autres. Les deux parties essentielles de notre gestion d'adresses sont donc maintenant :

1. Le programme
2. Le fichier

Organisation du fichier

Si nous demandions maintenant quelle partie de notre gestion d'adresses doit être organisée en premier, beaucoup répondraient "le programme". Mais c'est une erreur. Il faut en effet savoir d'abord ce que nous allons stocker, comment et où nous allons le stocker. Nous avons déjà décidé du "comment" puisque nous avons choisi le fichier séquentiel. Il nous faut maintenant définir où va se faire le stockage. Il existe deux possibilités :

1. Le stockage sur cassette
2. Le stockage sur disquette

Nous stockerons nos données sur le lecteur de disquette intégré puisque c'est le moyen de stockage le plus puissant et le plus rapide et qu'en outre, par définition, tous nos lecteurs le possèdent.

Reste la question de savoir ce que nous allons stocker. La réponse qui vient immédiatement à l'esprit c'est que ce sont

évidemment des adresses que nous voulons stocker. Encore faut-il savoir exactement ce que représente une adresse. Voyons donc de quels éléments se compose en principe une adresse :

- Titre	- Code postal
- Prénom	- Localité
- Nom	- Numéro de téléphone
- Rue	- Remarque

Ces divers éléments constitutifs d'une adresse sont appelés des CHAMPS. On parlera par exemple du champ 'Nom'. La totalité de ces champs constitue un enregistrement. Dans notre exemple, un enregistrement se composera donc de 8 champs. L'ensemble constitué par tous les enregistrements est le fichier. Le tableau suivant illustre cette structure d'un fichier :

```
-------------------------
CHAMP 1  CHAMP 2  CHAMP 3  ... CHAMP n  ENREGISTREMENT 1
-------------------------
CHAMP 1  CHAMP 2  CHAMP 3  ... CHAMP n  ENREGISTREMENT 2
-------------------------
CHAMP 1  CHAMP 2  CHAMP 3  ... CHAMP n  ENREGISTREMENT 3
-------------------------
CHAMP 1  CHAMP 2  CHAMP 3  ... CHAMP n  ENREGISTREMENT n
--------- FICHIER -------------
```

La hiérarchie est donc FICHIER - ENREGISTREMENT - CHAMP

STOCKAGE DES DONNEES A L'INTERIEUR DE L'ORDINATEUR

Le travail avec un fichier séquentiel est très aisé mais ce type de fichier présente aussi des inconvénients par rapport à d'autres formes d'organisation de fichiers. Supposez que votre fichier complet d'adresses figure sur disquette et que vous vouliez maintenant une adresse bien précise qui figure sur ce fichier. C'est là que surgit le problème que posent les fichiers séquentiels. En effet, vous ne pouvez pas lire uniquement un enregistrement d'un fichier sans lire la totalité du fichier. Le

lecteur de disquette ne peut pas lire des enregistrements isolés de fichiers séquentiels. Comment faut-il donc faire pour accéder à une adresse déterminée ?

Pour pouvoir accéder ne serait-ce qu'à un seul enregistrement, le fichier tout entier doit être lu et placé dans la mémoire de l'ordinateur. Un fichier séquentiel ne doit par conséquent pas être plus grand que la capacité de mémoire disponible de l'ordinateur. Cette méthode présente cependant également des avantages. En effet, lorsque le fichier est entièrement en mémoire, l'accès aux différents enregistrements du fichier peut se faire instantanément. Le déroulement du programme suivra donc les étapes suivantes :

1. Chargement du fichier
2. Lecture, modification, suppression des enregistrements
3. Sauvegarder le fichier

En début de programme, il faudra donc d'abord charger en mémoire le fichier tout entier. Ensuite les divers enregistrements pourront être traités. Avant la fin du programme, il faudra encore sauvegarder à nouveau le fichier, du moins s'il a été modifié, c'est-à-dire si certains enregistrements ont été modifiés ou supprimés ou si d'autres ont été ajoutés.

Les variables

Nous avons donc dit qu'il fallait stocker les enregistrements dans l'ordinateur. Mais où et comment les stocker ?

LES DONNEES SONT STOCKEES A L'INTERIEUR DE L'ORDINATEUR DANS DES VARIABLES

Les variables sont des zones de la mémoire du CPC 6128 Plus qui reçoivent un nom qui permet d'accéder à ces zones de mémoire. Nous connaissons déjà deux types de données, les données numériques (les nombres) et les données alphanumériques (les chaînes de caractères ou texte). Les variables alphanumériques se distinguent par le signe '$'

apposé à la suite du nom d'une variable. Les noms des variables ne peuvent comporter plus de 40 caractères et ils doivent toujours commencer par une lettre. Les autres caractères composant le nom d'une variable peuvent être des chiffres ou des lettres. Les instructions ou les fonctions Basic peuvent faire partie du nom d'une variable mais elles ne peuvent être utilisées telles quelles pour désigner une variable, par exemple

```
PRINT, PI, KEY, etc...
```

Exemples de variables numériques :

```
COMPTEUR
TELEPHONE
MONTANT
```

Exemples de variables alphanumériques :

```
NOM$
DESIGNATIONARTICLE$
NUMERODECOMPTE$
X$
```

Il est vivement recommandé de choisir pour vos variables des noms suffisamment explicites, par exemple PRENOM$ pour le prénom ou PRIXDEREVIENT pour le prix de revient. Des noms de variables explicites contribuent considérablement à améliorer la lisibilité et la clarté du listing d'un programme.

Traitement des variables

Revenons enfin à notre ordinateur. Nous allons maintenant créer, traiter et sortir des variables.

Le stockage de données numériques dans des variables est un véritable jeu d'enfants. Si vous voulez par exemple stocker 45.12 dans la variable PRIXDEREVIENT, l'instruction à entrer sera :

```
PRIXDEREVIENT=45.12
```

Si vous voulez maintenant sortir le contenu de cette variable sur l'écran, il vous suffira d'entrer :

```
PRINT PRIXDEREVIENT
```

Essayez ces deux instructions. Pour supprimer maintenant la variable PRIXDEREVIENT, il faudra entrer :

```
PRIXDEREVIENT=0
```

Les variables peuvent également être utilisées dans des calculs. Essayons par exemple de faire sortir le double de la valeur de PRIXDEREVIENT. Si vous avez supprimé cette variable, entrez à nouveau 'PRIXDEREVIENT=45.12'. Vous pouvez alors entrer :

```
PRINT PRIXDEREVIENT*2
```

L'ordinateur multiplie par deux la valeur de PRIXDEREVIENT puis sort le résultat (90.24). La variable conserve cependant sa valeur de 45.12.

Comment faut-il donc procéder si vous voulez que la valeur de PRIXDEREVIENT double effectivement ? Comme vous voulez affecter dans ce cas une valeur nouvelle à cette variable, l'instruction qui convient est 'PRIXDEREVIENT='. L'instruction complète pour doubler le contenu de cette variable est :

```
PRIXDEREVIENT=PRIXDEREVIENT*2
```

L'ordinateur calcule toujours d'abord la valeur de l'expression placée à droite du signe égale puis il affecte ensuite le résultat à la variable placée à gauche du signe égale.

Les variables alphanumériques peuvent être créées de façon analogue. Plaçons par exemple le mot PARIS dans la variable VILLE$. L'instruction sera :

```
VILLE$="PARIS"
```

L'important est de ne pas oublier les guillemets. Tout texte sans guillemets est en effet considéré par l'ordinateur non pas

comme une chaîne de caractères mais comme le nom d'une variable. Les variables alphanumériques peuvent être également sorties sur l'écran avec l'instruction PRINT. Voyez vous-même :

```
PRINT VILLE$
```

Vous pouvez également supprimer une variable alphanumérique en lui affectant ce qu'on appelle une chaîne vide :

```
VILLE$=""
```

Une chaîne vide se compose donc de deux guillemets consécutifs. La différence par rapport aux variables numériques est que bien sûr l'ordinateur ne peut effectuer de calculs avec des variables alphanumériques, même si ces variables ne contiennent que des chiffres comme par exemple X$="123". L'ordinateur ne pourra effectuer aucun calcul avec X$.

Les chaînes peuvent cependant être assemblées ou "concaténées" comme le montre la suite d'instructions suivante :

```
VILLE1$="LYON"
VILLE2$="PARIS"
VILLE$=VILLE2$+VILLE1$
PRINT VILLE$
```

Les noms des deux villes ont d'abord été stockés dans les deux variables VILLE1$ et VILLE2$. Puis une troisième chaîne de caractères a été constituée à partir de l'assemblage des deux autres variables alphanumériques. La quatrième instruction a alors sorti sur l'écran la variable VILLE$.

En fait, il serait plus élégant de placer un trait d'union entre les deux villes. Il faudrait pour cela modifier la troisième instruction ainsi :

```
VILLE$=VILLE2$+"-"+VILLE1$
```

Nous connaissons maintenant les deux types de variables les plus importants, les variables numériques et les variables alphanumériques marquées par le signe $. Il est tout à fait possible de stocker des adresses dans de telles variables. Mais pour stocker plusieurs enregistrements de données possédant la même structure, il existe une encore meilleure possibilité.

Les tableaux

Les tableaux sont utilisés très fréquemment dans les programmes de traitement de données. Lorsque vous stockez dans l'ordinateur plusieurs données appartenant à un même groupe, il est pénible de devoir donner un nom différent à chaque élément de ce groupe. Il est en effet possible de donner à divers éléments d'un même groupe le même nom. Prenons l'exemple d'un groupe de cinq noms de villes :

```
BONN
PARIS
LONDRES
ROME
MADRID
```

Si nous voulions stocker ces cinq noms de villes dans la mémoire de l'ordinateur, il nous faudrait normalement employer cinq noms de variables différents alors qu'un seul nom suffirait si nous plaçons ces noms dans un tableau (array en anglais). Comme il faut cependant pouvoir distinguer entre eux les différents éléments de notre groupe, un numéro sera ajouté entre parenthèses à la suite du nom de la variable. Ce numéro s'appelle un indice. Le premier élément du tableau recevra donc l'indice 1, le second l'indice 2, etc... En fait, le premier indice possible est 0 mais nous négligerons cette possibilité qui ne peut que compliquer les choses. Notez toutefois que si vous n'utilisez pas l'indice 0, vous gaspillez un peu de place en mémoire.

Une variable-tableau sera par exemple ADRESSE$(1) ou XY(212). Vous pouvez choisir n'importe quel nombre comme indice, en respectant cependant les deux règles suivantes :

1. Si l'indice dépasse 10, il faut réserver de la place en mémoire pour le tableau correspondant.

2. Le tableau ne doit pas dépasser la capacité mémoire de l'ordinateur.

Les tableaux sont donc gérés automatiquement par l'ordinateur jusqu'à l'indice 10. Mais si l'indice d'un tableau dépasse 10, il faut définir ce tableau, le "dimensionner" avec l'instruction DIM.

Problème :	Dimensionnement des tableaux	
Instruction :	DIM nom(n1,n2,n3...)	
Paramètres :	nom n1,n2	nom de la variable-tableau valeur maximale de l'indice pour chaque dimension
Exemple :	DIM D$(20) Le tableau D$ sera créé avec un indice pouvant aller jusqu'à 20	
Remarque :	Chaque tableau ne peut être dimensionné qu'UNE SEULE FOIS dans un programme, sinon vous obtenez le message d'erreur "Array already dimensioned"	

Si l'indice maximum est par exemple 5, il n'est pas nécessaire de dimensionner votre tableau. Le seul petit inconvénient est cependant que, si vous ne dimensionnez pas un tableau dont l'indice maximum est 5, l'ordinateur réservera automatiquement de la place pour un tableau de 10 éléments (11 avec l'indice 0). La place réservée pour les éléments 6 à 10 sera donc gaspillée. C'est pourquoi vous pouvez malgré tout, pour économiser de la place en mémoire, dimensionner un tableau avec une instruction DIM, même si l'indice maximum est inférieur à 10.

Mais revenons maintenant à nos 5 noms de villes que nous voulions placer dans un tableau. Si nous utilisons la variable-

tableau VILLE$, les noms de villes pourront être stockés avec les instructions suivantes :

```
10 VILLE$(1)="BONN"
20 VILLE$(2)="PARIS"
30 VILLE$(3)="LONDRES"
40 VILLE$(4)="ROME"
50 VILLE$(5)="MADRID"
```

Nous aurions pu auparavant entrer l'instruction 'DIM VILLE$(5)', si nous étions sûr que les indices 6 à 10 ne seraient pas utilisés.

Si vous utilisez un indice qui excède les dimensions d'un tableau, vous obtenez le message d'erreur "Subscript out of range".

Nous allons maintenant, dans l'exemple suivant, placer dans un tableau les noms de pays correspondant aux capitales que nous venons de stocker. Ce tableau recevra le nom PAYS$. Voici les instructions nécessaires :

```
 60 PAYS$(1)="RFA"
 70 PAYS$(2)="FRANCE"
 80 PAYS$(3)="ANGLETERRE"
 90 PAYS$(4)="ITALIE"
100 PAYS$(5)="ESPAGNE"
```

Nous avons donc ainsi créé deux tableaux dont les indices se correspondent puisque VILLE$(1) contient le nom de la capitale de PAYS$(1). D'une manière générale, VILLE$(X) contient la capitale de PAYS$(X).

Le principal intérêt des tableaux vient du fait que l'on peut utiliser comme indice, non seulement un nombre, mais aussi une variable ou une expression quelconque. Il est en effet parfaitement possible de définir ainsi l'indice d'un tableau : D$(X-2*I+13). Les avantages de l'indiciation sont donc évidents.

Les deux tableaux que nous venons de créer sont des tableaux à une dimension car ils ne possèdent qu'un indice. Mais un

tableau peut comporter autant d'indices que vous le souhaitez. Au lieu des deux tableaux à une dimension que nous venons de créer, nous pourrions par exemple avoir un tableau à deux dimensions. Examinez d'abord la suite d'instructions suivante :

```
10 DIM TABLEAU$(5,1)
20 TABLEAU$(1,0)="BONN"
30 TABLEAU$(2,0)="PARIS"
40 TABLEAU$(3,0)="LONDRES"
50 TABLEAU$(4,0)="ROME"
60 TABLEAU$(5,0)="MADRID"
70 TABLEAU$(1,1)="RFA"
80 TABLEAU$(2,1)="FRANCE"
90 TABLEAU$(3,1)="ANGLETERRE"
100 TABLEAU$(4,1)="ITALIE"
110 TABLEAU$(5,1)="ESPAGNE"
```

L'instruction DIM en ligne 10 n'est pas indispensable mais elle évite que l'ordinateur n'exécute automatiquement l'instruction DIM TABLEAU$(10,10) qui réserverait 121 places en mémoire alors que nous n'avons besoin de place que pour 10 éléments de tableau. Vous voyez que pour les tableaux à plusieurs dimensions, il vaut tout de même mieux indiquer précisément la place que l'ordinateur doit réserver car sinon vous risquez de consommer inutilement beaucoup de place en mémoire.

Nous avons maintenant un tableau à deux dimensions dont le second indice sert à identifier le type de donnée (ville ou pays). On peut représenter ce tableau sous la forme d'une matrice :

	0	1
1	BONN	RFA
2	PARIS	FRANCE
3	LONDRES	ANGLETERRE
4	ROME	ITALIE
5	MADRID	ESPAGNE

Revenons maintenant à notre gestion d'adresses. Un tableau à deux dimensions tel que celui ci-dessus est idéal pour recevoir les données d'un fichier séquentiel. Le premier indice peut en effet correspondre au numéro des différents enregistrements et

le second indice au numéro des différents champs de chaque enregistrement. Comme nos enregistrements comprennent 8 champs, ce second indice vaudra donc au maximum 8. Reste à définir la valeur maximale du premier indice, c'est-à-dire le nombre maximum d'adresses. Nous supposerons que le nombre d'adresses à traiter est de 200. Le nom du tableau sera ADRESSE$. Il faudra donc entrer l'instruction suivante pour dimensionner ce tableau :

```
DIM ADRESSE$(200,8)
```

Nous verrons plus tard comment traiter ce tableau.

Entrée de données au clavier

Dans les exercices précédents, nous nous sommes contenté de traiter et de sortir des données. Pour aller plus loin, il nous faut maintenant entrer des données au clavier. L'instruction qui permet cela est INPUT. L'instruction INPUT nécessite deux paramètres. Une chaîne de caractères qui peut être sortie avant exécution de l'instruction INPUT proprement dite et une variable qui recevra la donnée qui sera entrée au clavier. L'instruction INPUT peut par ailleurs être utilisée aussi bien en mode direct qu'en mode programme.

Problème :	Entrée de données au clavier	
Instruction :	INPUT chaîne,v1	
Paramètres :	chaîne v1	un texte quelconque variable qui recevra la donnée entrée
Exemple :	INPUT"NOMBRE";X La chaîne de caractères "NOMBRE" sera sortie et une valeur numérique sera ensuite entrée au clavier et stockée dans la variable X.	
Remarque :	Cette instruction peut parfaitement être utilisée en mode direct.	

Entrez maintenant l'instruction NEW pour supprimer tout programme pouvant se trouver en mémoire. Les lignes Basic suivantes vous montrent un exemple d'application de l'instruction INPUT :

```
10 PRINT "CALCUL D'UN CERCLE"
20 PRINT "------------------"
30 INPUT "DIAMETRE ";diametre
40 PRINT "SURFACE DU CERCLE:";(diametre/2)^2*PI
```

Lancez ce programme avec l'instruction RUN. La ligne 30 demande le diamètre du cercle. Toute entrée doit être terminée en appuyant sur la touche [Return]. La chaîne de caractères doit absolument être suivie d'un point-virgule (ou d'une virgule, ce qui aura pour effet de ne pas afficher le point d'interrogation), mais la chaîne de caractères elle-même est facultative ainsi que le montre le programme suivant :

```
10 PRINT "CALCUL D'UN CERCLE"
20 PRINT "------------------"
30 INPUT diametre
40 PRINT "SURFACE DU CERCLE:";(diametre/2)^2*PI
```

Pour modifier la première version de notre programme, souvenez-vous que vous n'avez pas à taper à nouveau les 4 lignes de programme et qu'il suffit que vous modifiez la ligne 30. Comme vous le voyez, la chaîne de caractères n'a dans INPUT qu'une fonction de commentaire. Ce commentaire peut cependant être également confié à une instruction PRINT avant INPUT :

```
10 PRINT "CALCUL D'UN CERCLE"
20 PRINT "------------------"
25 PRINT "DIAMETRE ";
30 INPUT diametre
40 PRINT "SURFACE DU CERCLE:";(diametre/2)^2*PI
```

Si vous entrez la ligne 25 ci-dessus et que vous lancez ensuite le programme, vous constaterez qu'il fonctionne exactement comme la première version. L'important est le point-virgule à la

suite du PRINT en ligne 25. Si vous l'omettez, l'instruction INPUT sera exécutée sur la ligne suivante de l'écran. Essayez-le.

L'instruction INPUT permet également d'entrer des chaînes de caractères au clavier. Voici un exemple que nous vous invitons à entrer après avoir supprimé le programme précédent avec NEW :

```
10 PRINT"COMMENT VOUS APPELEZ-VOUS";
20 INPUT NOM$
30 PRINT "BONJOUR ";NOM$;", COMMENT ALLEZ-VOUS?"
```

Les instructions des lignes 10 et 20 peuvent également être concentrées dans une seule instruction INPUT qui sortira également le message :

```
10 INPUT"COMMENT VOUS APPELEZ-VOUS";NOM$
20 PRINT "BONJOUR ";NOM$;", COMMENT ALLEZ-VOUS?"
```

Remarquez que nous n'avons pas placé de point d'interrogation dans les questions en ligne 10 car INPUT ajoute automatiquement un point d'interrogation. Notez également que si vous appuyez simplement sur [Return] sans avoir rien entré, la variable sera annulée, c'est-à-dire qu'elle se verra affecter 0 pour une variable numérique et "" pour une variable alphanumérique.

Les boucles

Faisons d'abord un exercice qui nous amènera ensuite à expliquer les boucles. Nous voulons entrer cinq noms au clavier que nous stockerons dans le tableau NOM$. Avec les instructions que nous connaissons jusqu'à présent, nous pourrions construire le programme suivant :

```
10 PRINT"ENTREE DES NOMS"
20 PRINT"---------------"
30 INPUT"NOM ";NOM$(1)
40 INPUT"NOM ";NOM$(2)
50 INPUT"NOM ";NOM$(3)
```

```
60 INPUT"NOM ";NOM$(4)
70 INPUT"NOM ";NOM$(5)
```

Ce programme nous permet bien d'entrer cinq noms. Mais est-ce que la fonction principale de l'informatique n'est pas justement d'éviter les tâches répétitives ! Supposez que nous ayons ainsi 100 noms à entrer !

C'est là qu'interviennent les boucles. Voyons tout de suite la description des instructions qui régissent les boucles :

Problème :	Commande d'une boucle	
Instruction :	FOR v1=n1 TO n2 STEP n3	
Paramètres :	v1	variable numérique
	n1	valeur initiale de v1
	n2	valeur finale de v1
	n3	incrément, valeur de progression de v1 après chaque parcours de la boucle
Exemple :	FOR A=1 TO 20 STEP 1 La variable A sera augmentée de 1 après chaque parcours de la boucle, jusqu'à ce qu'elle atteigne 20. La boucle sera donc parcourue 20 fois.	
Remarque :	Si l'incrément est 1, l'instruction STEP et son paramètre peuvent être négligés. L'instruction donnée en exemple peut donc se présenter également sous la forme : FOR A=1 TO 20	

Toute boucle commence par une instruction FOR et se termine par une instruction NEXT dont nous allons maintenant voir la description. Toutes les instructions placées après le début de la boucle (l'instruction FOR) seront donc exécutées jusqu'à ce que le programme rencontre la fin de la même boucle (l'instruction NEXT correspondante). Arrivé à l'instruction NEXT, l'ordinateur examine si la valeur de la variable de la boucle a déjà dépassé la valeur finale qui lui a été fixée. Tant que ce n'est pas le cas, cette variable est augmentée de l'incrément fixé par

l'instruction STEP et la boucle est à nouveau parcourue du début, c'est-à-dire que toutes les instructions entre FOR et NEXT sont à nouveau exécutées. Si la variable de la boucle a par contre déjà dépassé sa valeur finale, le programme se poursuit après l'instruction NEXT dont voici la description :

Problème :	Fixer la fin d'une boucle	
Instruction :	NEXT v1	
Paramètres :	v1	variable de la boucle
Exemple :	NEXT A Désigne la fin de la boucle ouverte avec l'instruction FOR A=... (comme dans l'exemple précédent)	
Remarque :	Si la variable v1 n'est pas indiquée, l'instruction NEXT se rapportera à la dernière boucle ouverte.	

Ces deux instructions nous permettent maintenant de programmer n'importe quelles boucles. Revenons maintenant à notre programme d'entrée de données avec INPUT. Puisqu'il s'agit d'entrer des données dans un tableau de l'indice 1 à l'indice 5, nous pouvons construire une boucle qui nous permettra de n'utiliser qu'une seule instruction INPUT. Notre boucle devra parcourir les valeur 1 à 5. Elles se présentera donc ainsi :

```
FOR I=1 TO 5
```

Le programme complet sera donc maintenant :

```
10 PRINT"ENTREE DES NOMS"
20 PRINT"---------------"
30 FOR I=1 TO 5
40 INPUT"NOM ";NOM$(I)
50 NEXT I
```

Remarquez bien la logique de ce programme : les deux instructions PRINT en lignes 10 et 20 sont tout d'abord

exécutées puis la boucle avec la variable I commence, la valeur initiale étant 1, la valeur finale 5 et l'incrément 1. Souvenez-vous que l'incrément est toujours 1 lorsque l'instruction STEP est négligée. Au début de la boucle, la variable I reçoit la valeur 1. La ligne 40 permet alors l'entrée de l'élément du tableau d'indice 1. L'instruction NEXT examine ensuite si la fin de la boucle est atteinte, c'est-à-dire si la variable I a déjà atteint 5. Comme ce n'est pas encore la cas, I est augmenté de l'incrément 1 puis le programme se poursuit à partir de l'instruction placée après le FOR, donc à partir de la ligne 40. La variable I vaut maintenant 2. L'instruction INPUT suivante permet donc d'entrer l'élément du tableau d'indice 2, et ainsi de suite, jusqu'à ce que I atteigne 6 (=5+1). Lorsque le programme rencontrera alors à nouveau NEXT et que I vaudra 5, il ne retournera pas à la ligne 40 mais continuera l'exécution du programme après NEXT. En l'occurrence, le programme sera alors terminé.

Nous pouvons maintenant compléter notre programme pour que les noms soient sortis après avoir été tous entrés. Il nous faut pour cela ajouter une autre boucle :

```
60 FOR I=1 TO 5
70 PRINT"NOM ";NOM$(I)
80 NEXT I
```

Vous voyez que le mécanisme des boucles est très simple.

Que peut-on encore faire avec une boucle ? D'abord vous pouvez par exemple construire une boucle d'attente pour ralentir le déroulement de votre programme. Il suffit pour cela de placer à l'endroit où vous voulez ralentir votre programme, une boucle dont l'instruction FOR sera immédiatement suivie par l'instruction NEXT. Votre ordinateur peut parcourir une telle boucle environ 1000 fois en une seconde. Si vous voulez donc créer un délai de 3 secondes environ, vous pouvez utiliser la boucle suivante :

```
FOR X=1 TO 3000:NEXT X
```

Notez par la même occasion que vous pouvez parfaitement placer plusieurs instructions sur une même ligne, à condition

que ces instructions soient séparées entre elles par un double-point. Plaçons maintenant une boucle d'attente dans notre programme, entre l'entrée des données et leur sortie sur écran. Il nous faut pour cela écrire une ligne entre les deux parties de notre programme. Cette ligne peut donc recevoir le numéro 55 par exemple. Voici la ligne qu'il faut entrer pour avoir un délai de 2 secondes entre l'entrée et la sortie des données :

```
55 FOR X=1 TO 2000:NEXT X
```

Vous pouvez maintenant faire apparaître sur l'écran le programme ainsi complété en entrant l'instruction LIST. Entrez ensuite RUN et vous verrez l'effet de la ligne 55.

Nous n'avons jusqu'ici travaillé qu'avec un incrément de 1. Mais nous pouvons également faire entrer nos données "à l'envers", de l'indice 5 à l'indice 1. La valeur initiale de la variable de boucle sera alors 5, la valeur finale 1 et l'incrément -1. Il faut pour cela modifier les lignes 30 et 60 ainsi :

```
30 FOR I=5 TO 1 STEP -1
60 FOR I=5 TO 1 STEP -1
```

Faisons encore quelques exercices avec les boucles FOR ... NEXT. Essayez d'écrire les instructions FOR correspondant aux boucles suivantes :

Variable de boucle	Valeur initiale	Valeur finale	Incrément
A) INDICE	10	18	1
B) NOMBRE	30	15.5	-0.5
C) BOUCLE	590	1800	0.25
D) TEMPS	1	10	0.3

Les solutions :

```
A)    FOR INDICE=10 TO 18 STEP 1
ou
      FOR INDICE=10 TO 18
B)    FOR NOMBRE=30 TO 15.5 STEP -0.5
C)    FOR BOUCLE=590 TO 1800 STEP 0.25
D)    FOR TEMPS=1 TO 10 STEP 0.3
```

Encore une remarque pour en terminer avec la présentation des boucles : si vous construisez des boucles dont la logique est fausse, par exemple FOR I=10 TO 0, ces boucles ne seront pas parcourues une seule fois. L'ordinateur ignorera donc toutes les instructions placées entre FOR et NEXT.

Premières réactions du programme

Nous avons maintenant suffisamment avancé dans l'étude du Basic pour que nous puissions nous attaquer enfin à notre programme d'adresses. Si vous avez déjà utilisé des programmes, vous savez que tout programme commence par un en-tête qui indique le nom du programme. C'est cet en-tête que nous allons maintenant réaliser. Entrez donc les lignes suivantes telles que nous vous les proposons, sans vous inquiéter si vous voyez des instructions que nous n'avons pas encore décrites. Nous y viendrons plus tard.

```
100 REM ========================
110 REM    EN-TETE DU PROGRAMME
120 REM ========================
130 CLS
140 PRINT STRING$(40,"=")
150 LOCATE 12,2
160 PRINT "GESTION D'ADRESSES"
170 PRINT STRING$(40,"=")
```

Après avoir entré et lancé cette section de programme, vous aimeriez certainement bien savoir ce que signifient les nouvelles instructions. Commençons par les lignes 100-120 qui

servent de commentaire du programme, pour le programmeur. Afin que l'ordinateur n'interprète pas le texte de ces commentaires comme des instructions, nous avons placé l'instruction REM au début de chacune de ces lignes. REM signifie REMarque. Vous pouvez placer des lignes de commentaire commençant par REM en n'importe quel endroit de votre programme. Les commentaires rendent la lecture et la compréhension du listing d'un programme plus aisées.

La deuxième nouveauté n'est pas une instruction mais une fonction et plus précisément une fonction de chaîne de caractères (string en anglais). La fonction STRING$ permet de constituer une chaîne composée d'une répétition d'un même caractère :

Problème :	Constituer une chaîne de caractères	
Instruction :	STRING$(n,"z")	
Paramètres :	n "z"	nombres de caractères (de 0 à 255) caractère à reproduire
Exemple :	PRINT STRING$(80,":") 80 double-points seront sortis	

Vous pouvez tester cette fonction à votre guise en l'employant avec l'instruction PRINT. Comme cette instruction est très simple, passons à la nouveauté suivante.

L'instruction PRINT sort des données sur l'écran. L'endroit de l'écran où se fait cette sortie dépend de l'emplacement du curseur qui dépend lui-même en principe de la dernière instruction PRINT utilisée, ou de l'instruction CLS. Mais vous pouvez également déterminer arbitrairement où la prochaine sortie avec PRINT devra se produire, grâce à l'instruction suivante :

Problème :	Positionnement du curseur

Instruction :	LOCATE colonne,ligne	
Paramètres :	colonne	nouvelle colonne du curseur
	ligne	nouvelle ligne du curseur
Exemple :	LOCATE 5,10 place le curseur en colonne 5 ligne 10	

Remarquez que le curseur est invisible lors de l'exécution d'un programme. Sa fonction reste toutefois la même et l'ordinateur sait parfaitement où il se trouve à tout moment. L'instruction LOCATE utilisée avant une instruction PRINT permet donc de déterminer à quel endroit se fera une sortie sur écran. Supposons par exemple que nous voulions sortir le texte 'BONJOUR' au milieu de l'écran. Nous utiliserions l'instruction suivante :

```
LOCATE 18,13:PRINT"BONJOUR"
```

Rappelez-vous que vous pouvez placer plusieurs instructions sur une même ligne, à condition de les séparer entre elles par des double-points.

Revenons maintenant à notre programme.

Les sous-programmes

Les lignes Basic que nous venons d'entrer pour éditer l'en-tête du programme constituent une partie de programme dont nous aurons souvent besoin. Pour nous éviter d'avoir à taper ces lignes autant de fois que nous voulons les utiliser, nous allons les transformer en un sous-programme. Mais comment définir un sous-programme et comment le faire exécuter ?

Nous avons déjà rencontré au chapitre précédent l'instruction GOTO qui permet de faire sauter le programme d'une ligne à une autre. Mais nous avons besoin d'une instruction qui saute à un sous-programme tout en se rappelant d'où le programme a sauté au sous-programme, de façon à ce que, une fois le sous-

programme exécuté, le programme reprenne son cours normal, à partir de l'endroit d'où il a sauté au sous-programme. Cette instruction, qui est de la même famille que GOTO, est GOSUB dont voici la description :

Problème :	Appel d'un sous-programme
Instruction :	GOSUB numéro de ligne
Paramètres :	numéro de ligne numéro de la ligne où commence le sous-programme
Exemple :	GOSUB 100 saute au sous-programme commençant en ligne 100
Remarque :	- Les sous-programmes appelés avec GOSUB doivent se terminer par l'instruction Return. - Si le programme rencontre une instruction Return sans qu'un sous-programme n'ait été appelé, l'ordinateur envoie le message d'erreur 'Unexpected Return'.

Un sous-programme est donc très simple à réaliser. Dans notre exemple, pour transformer le début de notre programme en un sous-programme que nous ferons appeler par la suite, il suffit d'ajouter la ligne suivante aux lignes que nous avons déjà entrées :

```
180 RETURN
```

Mais nous ne pouvons pas lancer notre programme sous sa forme actuelle. En effet, lorsque le programme rencontrera l'instruction Return en ligne 180, il nous enverra le message d'erreur 'Unexpected Return in 180', ce qui est normal puisque notre sous-programme doit absolument être appelé avec l'instruction GOSUB.

Mais voyons maintenant comment nous allons organiser les numéros de ligne de notre programme :

0-99 nous placerons ici les préparatifs du programme, par exemple le dimensionnement des tableaux ou la définition de variables
100-999 nous placerons ici tous les sous-programmes
1000- le programme proprement dit, le programme principal commencera en ligne 1000

Le programme devra donc exécuter d'abord la partie composée des lignes 0 à 99 puis passer au corps du programme en ligne 1000 d'où seront appelés au fur et à mesure des besoins les sous-programmes placés dans la partie composée des lignes 100 à 999. Il faut donc sauter la partie réservée au sous-programme. Entrons donc la ligne suivante :

```
99 GOTO 1000
```

Nous pouvons maintenant placer en lignes REM le titre du programme principal, puis appeler ensuite immédiatement le sous-programme d'en-tête du programme :

```
1000 REM ========================
1010 REM    PROGRAMME PRINCIPAL
1020 REM ========================
1030 GOSUB 100
```

Vous pouvez essayer maintenant notre programme qui fonctionnera sans erreur. Il édite l'en-tête du programme.

Le menu

Voyons maintenant à quel point notre programme est arrivé :

```
99 GOTO 1000
100 REM ========================
110 REM    EN-TETE DU PROGRAMME
120 REM ========================
130 CLS
140 PRINT STRING$(40,"=")
150 LOCATE 12,2
160 PRINT "GESTION D'ADRESSES"
```

```
170 PRINT STRING$(40,"=")
180 RETURN
1000 REM ==========================
1010 REM    PROGRAMME PRINCIPAL
1020 REM ==========================
1030 GOSUB 100
```

Avant de passer à la partie spécifique, de notre programme, il faut que nous définissions clairement les tâches que nous voulons confier à notre programme. Tout d'abord, il faut que le programme puisse stocker des données sur un périphérique de stockage et qu'il puisse ensuite les recharger à partir de ce même périphérique. Si notre programme ne remplissait pas au moins ces deux fonctions, il serait sans intérêt. Nous devons donc pouvoir disposer des deux options suivantes :

```
-1- CHARGER LE FICHIER
-2- SAUVEGARDER LE FICHIER
```

Il faut bien sûr également que nous puissions entrer des adresses et donc que nous ayons l'option suivante :

```
-3- ENTRER DES ADRESSES
```

Il faut également que nous puissions modifier les données du fichier, d'où l'option 4 :

```
-4- MODIFIER DES ADRESSES
```

Si vous vous fâchez à mort avec votre oncle Anatole, il faut aussi que vous puissiez le supprimer de votre fichier, d'où l'option 5 :

```
-5- SUPPRESSION D'ADRESSES
```

Enfin, le principal : les données ne nous servent à rien si nous ne pouvons pas les consulter. L'option suivante permet donc la sortie des adresses :

```
-6- SORTIR LES ADRESSES
```

Comme il n'est pas très élégant d'arrêter un programme soit avec la touche [ESC], soit en coupant l'ordinateur, ajoutons un dernier point :

```
-7- FIN DU PROGRAMME
```

Maintenant que nous avons défini toutes les tâches qu'accomplira notre programme, nous pouvons construire le menu, c'est-à-dire la partie du programme qui permettra à l'utilisateur de choisir parmi les différentes options que nous avons prévues. Voici les lignes correspondant au menu :

```
1040 LOCATE 1,7
1050 PRINT"     FONCTIONS DU PROGRAMME:"
1060 PRINT"     ----------------------"
1070 PRINT
1080 PRINT"    -1- CHARGER LE FICHIER
1090 PRINT"    -2- SAUVEGARDER LE FICHIER
1100 PRINT"    -3- ENTRER DES ADRESSES
1110 PRINT"    -4- MODIFIER DES ADRESSES
1120 PRINT"    -5- SUPPRESSION D'ADRESSES
1130 PRINT"    -6- SORTIR LES ADRESSES
1140 PRINT"    -7- FIN DU PROGRAMME
```

Si vous lancez le programme maintenant, vous verrez qu'il commence à prendre lentement mais sûrement un aspect professionnel. Il faut maintenant que l'utilisateur puisse choisir une des fonctions du programme. Nous lui demanderons donc d'entrer une valeur numérique comprise entre 1 et 7, suivant la fonction choisie. Nous ferons sortir auparavant le texte "CHOIX".

```
1150 LOCATE 10,18:PRINT"Choix ";
1160 INPUT fonction
```

Test avec IF

Mais que faire si l'utilisateur entre un nombre incorrect, par exemple 9 ? Il faut donc que nous puissions tester si la valeur entrée correspond bien aux valeurs possibles. L'instruction qui nous le permet est IF, dont voici la description :

Problème :	Tester si une condition est remplie	
Instruction :	IF condition THEN instruction	
Paramètres :	condition	comparaison (par exemple A=10 ou B=12)
	instruction	instruction qui devra être exécutée si la condition est vraie
Exemple :	IF F=0 THEN GOTO 1000 Si la variable F vaut 0, le programme saute à la ligne 1000, sinon le déroulement du programme se poursuit à partir de la ligne suivante.	
Remarque :	Si l'instruction qui suit THEN est un GOTO, le mot GOTO peut être négligé. Notre exemple aurait donc pu s'écrire également : IF F=0 THEN 1000	

La condition peut être définie à l'aide des opérateurs =, < ou >, ou d'une combinaison de ces opérateurs. Voici un tableau de tous les opérateurs de comparaison qui peuvent être employés dans un test :

Symbole	**Signification**
=	égale
>	supérieur à
<	inférieur à
>= ou =>	supérieur ou égal
<= ou <=	inférieur ou égal
<>	différent de

L'instruction IF contrôle donc d'abord si la condition est vraie. Si c'est bien le cas, les instructions placées à la suite de THEN sont exécutées, sinon le programme se poursuit à la ligne suivant la ligne avec l'instruction IF.

Cependant la puissance de cette instruction est encore accrue par le fait que plusieurs conditions peuvent être reliées avec des

opérateurs logiques. Voyons tout de suite concrètement ce que cela signifie :

```
IF REPONSE=1 OR TEST>12 THEN 1000
```

Nous avons ici, dans la condition, deux opérateurs reliés entre eux. L'opérateur OR (ou) est un ou inclusif, ce qui veut dire que : si REPONSE égale 1 OU si TEST est supérieur à 12, le programme doit sauter en 1000. Si l'une de ces deux conditions ou si les deux conditions sont remplies, le programme sautera à la ligne 1000. Il ne passera à l'exécution de la ligne suivante que si aucune des deux conditions n'est remplie. OR n'est qu'un des opérateurs logiques possibles. Voici un autre exemple :

```
IF REPONSE=1 AND TEST>12 THEN 1000
```

Avec AND (ET), la condition n'est remplie que si les deux conditions sont remplies, que si REPONSE égale 1 et si, en même temps, TEST est supérieur à 12.

Dans le problème qui nous occupe, nous voulions contrôler que le nombre entré par l'utilisateur soit bien compris entre 1 et 7 inclus. L'instruction IF correspondante sera donc :

```
IF fonction=0 OR fonction>7 THEN ...
```

Oui, mais justement, que doit-il se passer si le nombre entré par l'utilisateur sort du cadre voulu ? Eh bien nous enverrons un message d'erreur à l'utilisateur et nous reviendrons à la ligne 1000. Pour sortir le message d'erreur, nous avons besoin d'un autre sous-programme qui sorte le message que nous aurons placé dans une chaîne de caractères :

```
200 REM =====================
210 REM    MESSAGE D'ERREUR
220 REM =====================
230 LOCATE 1,25
240 PRINT erreur$
250 PRINT CHR$(7);
260 FOR I=1 TO 1000:NEXT I
270 LOCATE 1,25
```

```
280 PRINT STRING$(39," ")
290 RETURN
```

La ligne 230 place d'abord le curseur sur la dernière ligne. La ligne 240 sort la chaîne erreur$ dont le contenu a été défini par le programme d'appel : avant que le programme principal n'appelle ce sous-programme, il a donc affecté le message qui convient à la variable erreur$.

Code ASCII

Venons-en maintenant à la fonction de la ligne 250 que nous avions déjà utilisée avec les touches de fonction. Souvenez-vous que nous avions ajouté RETURN aux instructions, sous la forme du code CHR$(13). Il est temps d'expliquer comment nous avions procédé. Chaque caractère est géré par l'ordinateur sous la forme d'un code. En effet un ordinateur sans ses périphériques ne traite que des nombres. Chaque caractère mis en mémoire dans l'ordinateur reçoit donc un code compris entre 0 et 255. Ce codage suit généralement le standard américain ASCII dont les codes du CPC 6128 Plus ne s'écartent pratiquement pas. Le codage ne s'applique cependant pas uniquement aux caractères mais également à certaines fonctions spéciales, telles que le bip par exemple. Deux fonctions permettent de travailler avec les codes ASCII. En voici la description :

Problème :	Transformation d'un code ASCII en un caractère	
Instruction :	CHR$(n)	
Paramètres :	n	code ASCII (entre 0 et 255)
Exemple :	FOR I=32 TO 255:PRINT CHR$(I):NEXT I Montre tous les caractères pouvant être représentés par le CPC 6128 Plus	

Remarque :	- Si vous fournissez un code ASCII supérieur à 255, vous obtenez le message d'erreur "Improper argument" - Les codes ASCII inférieurs à 32 sont des codes de contrôle qui ne produisent aucun caractère affichable à l'écran.

Problème :	Transformation d'un caractère en un code ASCII	
Instruction :	ASC("c")	
Paramètres :	"c"	un caractère ou une variable alphanumérique comprenant au moins un caractère
Exemple :	PRINT ASC("A") Montre à l'écran le code ASCII de la lettre "A".	
Remarque :	- Si vous fournissez pour paramètre une chaîne vide, vous obtenez le message d'erreur "Improper argument" - Si la chaîne comporte plusieurs caractères, c'est le code du premier qui est fourni	

Vous pouvez donc apprendre, en mode direct, de l'ordinateur lui-même le code ASCII de n'importe quel caractère en entrant PRINT ASC("X").

La ligne 250 fait donc émettre un bip avec PRINT CHR$(7), de façon à prévenir l'utilisateur qu'il a commis une erreur. Il s'agit en l'occurrence d'un code standard, identique sur la plupart des ordinateurs.

Revenons à notre sous-programme d'erreur. Le signal sonore est suivi d'une boucle d'attente qui crée un délai d'environ une seconde pour permettre la lecture du message. Le message d'erreur est ensuite effacé en ligne 280 par 39 espaces. La ligne 290 conclut le sous-programme par un retour au programme principal.

Nous pouvons maintenant compléter notre test avec IF à la suite de INPUT :

```
1170 IF fonction=0 OR fonction>7 THEN erreur$="Valeur
incorrecte":GOSUB 200:GOTO 1150
```

Entrez maintenant cette instruction sur une ligne. L'utilisateur sera maintenant averti par le message d'erreur et le signal sonore sortis par le sous-programme d'erreur, chaque fois qu'il aura entré un nombre égal à 0 ou supérieur à 7. Faites-en vous-même le test en lançant le programme et en entrant une valeur incorrecte.

GOTO CALCULE

Si une valeur correcte a été entrée, il nous faut maintenant diriger le programme vers la partie du programme qui correspond au choix de l'utilisateur.

Nous allons utiliser pour cela l'instruction ON ... GOTO dont voici la description :

Problème :	Saut calculé	
Instruction :	ON v1 GOTO n1, n2, ...	
Paramètres :	v1 n1,n2	variable numérique numéros des lignes auxquelles le programme doit sauter en fonction de la valeur de v1
Exemple :	ON F GOTO 100,200,300,400 Suivant que F vaudra respectivement 1, 2, 3 ou 4, le programme sautera aux lignes 100, 200, 300 ou 400. Si F vaut 0 ou si F est supérieur à 4, il n'y aura pas de saut et le programme se poursuivra sur la ligne suivante.	

Remarque :	Si la variable v1 n'est pas un nombre entier (4.35 par exemple), seule la partie entière de v1 sera prise en compte

Nous avons bien là l'instruction idéale pour le problème qui nous est posé. Il nous reste à déterminer par quels numéros de ligne commenceront les différentes parties du programme et nous pouvons entrer l'instruction suivante :

```
1180 ON fonction GOTO 5000, 10000, 15000, 20000, 25000, 30000,
35000
```

Voici à quoi correspondent ces numéros de ligne :

'fonction'	Ligne	Partie du programme
1	5000	CHARGER LE FICHIER
2	10000	SAUVEGARDER LE FICHIER
3	15000	ENTRER LES ADRESSES
4	20000	MODIFIER LES ADRESSES
5	25000	SUPPRIMER DES ADRESSES
6	30000	SORTIR LES ADRESSES
7	35000	FIN DU PROGRAMME

Nous pouvons passer maintenant à la réalisation des différentes parties du programme, en commençant par la fonction 3, ENTRER LES ADRESSES. Mais auparavant, il nous faut dimensionner le tableau qui recevra les adresses. Ajoutons donc la ligne suivante :

```
10 DIM adresses$(200,7):REM tableau pour les adresses
```

Nous avons réservé de la place en mémoire pour 200 adresses, ce qui devrait être suffisant en principe pour une gestion d'adresses privée. Bien sûr ce programme pourrait plus difficilement être utilisé pour gérer les adresses d'une société.

Avant de réaliser les différentes parties du programme, il serait bien que nous écrivions encore un sous-programme (ou une routine) qui sorte un bel en-tête pour chaque partie du programme. Cette même routine (=sous-programme) sortira également auparavant l'en-tête du programme. Chaque partie de programme porte un nom qui est également indiqué dans le menu. Nous placerons donc les noms des différentes partie du programme dans un tableau que nous appellerons fonction$ et dont l'indice correspondra à la variable 'fonction'. Par exemple fonction$(3) contiendra le texte 'ENTRER LES ADRESSES' qui correspond au point 3 du menu. La mise en place de ce tableau relève des préparatifs; elle figurera donc dans la zone des lignes 0 à 99. Les lignes suivantes créent notre tableau :

```
20 fonction$(1)="  CHARGER LE FICHIER  "
21 fonction$(2)="SAUVEGARDER LE FICHIER"
22 fonction$(3)=" ENTRER LES ADRESSES  "
23 fonction$(4)="MODIFIER LES ADRESSES "
24 fonction$(5)="SUPPRIMER DES ADRESSES"
25 fonction$(6)="  SORTIR DES ADRESSES "
26 fonction$(7)="   FIN DU PROGRAMME   "
```

Faites attention à ce que toutes les chaînes aient bien la même taille (les guillemets doivent être placés les uns sous les autres). Le sous-programme placé en ligne 300 aura maintenant pour fonction de montrer l'en-tête du programme puis celui de la fonction choisie par l'utilisateur :

```
300 REM ================================
310 REM En-têtes des parties de programme
320 REM ================================
330 GOSUB 100
340 LOCATE 7,5:PRINT STRING$(24,"*")
350 LOCATE 7,6:PRINT "*" + fonction$(fonction) + "*"
360 LOCATE 7,7:PRINT STRING$(24,"*")
370 RETURN
```

Vous voyez ici qu'un sous-programme peut lui-même appeler un autre sous-programme. Notre sous-programme sort donc ainsi d'abord l'en-tête du programme principal puis la première ligne du cadre dans lequel s'inscrira le nom de la fonction. Le curseur est alors placé en colonne 7 ligne 6 où le nom de la fonction sera sorti entre deux étoiles. Ce nom est tiré du tableau fonction$ avec l'indice fonction. La ligne 360 ferme le cadre.

Cette routine devra donc être appelée lorsque l'utilisateur aura choisi une fonction, donc après le test IF. Ajoutons donc la ligne suivante :

```
1175 GOSUB 300
```

Il faut en effet que l'en-tête soit sorti avant que le programme ne saute à une des parties spécifiques du programme.

Entrer les adresses

Nous arrivons maintenant aux choses sérieuses. Commençons par le plus simple :

```
15000 REM ====================
15010 REM  ENTRER LES ADRESSES
15020 REM ====================
```

C'est maintenant que les choses se compliquent, mais ne perdez pas courage. Il faut que nous puissions trouver un moyen de compter le nombre d'enregistrements qui ont déjà été entrés. Nous augmenterons donc en permanence, dans cette partie du programme, un compteur qui contiendra le nombre d'adresses entrées. Entrons donc la ligne suivante :

```
15030 compteur=compteur+1
```

Si donc 'compteur' est non nul, c'est que des données auront déjà été entrées ou chargées.

Comme nous avons maintenant à faire entrer des données pour les 7 champs de chaque enregistrement, nous allons essayer

d'échapper aux 7 lignes INPUT qui seraient normalement nécessaires. Plaçons d'abord les intitulés des 7 champs dans un tableau (champ$(1) à champ$(7)) :

```
30 champ$(1)="TITRE      "
31 champ$(2)="PRENOM "
32 champ$(3)="NOM    "
33 champ$(4)="RUE    "
34 champ$(5)="VILLE  "
35 champ$(6)="TELEPHONE "
36 champ$(7)="REMARQUE   "
```

Comme nous avons ainsi placé les noms des champs dans un tableau en début de programme, nous pourrons ensuite les utiliser à n'importe quel moment du déroulement du programme. La section suivante du programme va vous permettre de comprendre à quoi ils nous serviront :

```
15040 PRINT
15050 FOR indice=1 TO 7
15060 PRINT champ$(indice);
15070 INPUT adresses$(compteur,indice)
15080 NEXT indice
```

Lancez le programme après avoir entré ces lignes, puis choisissez la fonction 3. Vous voyez que nous avons réussi grâce à l'emploi d'une boucle à faire entrer les données d'un enregistrement entier avec seulement 4 instructions. La boucle sort en effet d'abord le nom du champ en utilisant la variable 'indice' comme indice du tableau des noms de champs. Immédiatement après le nom du champ (point-virgule !), l'instruction INPUT permet l'entrée du contenu du champ. Nous utilisons ici deux indices car, souvenez-vous, notre tableau contenant les adresses est un tableau à deux dimensions. Le premier indice est fourni par la valeur de la variable 'compteur' qui est augmentée de 1 après l'entrée de chaque enregistrement. Comme second indice nous utilisons à nouveau la variable 'indice'. Nous permettons ainsi à l'utilisateur d'entrer au clavier le contenu des 7 champs que contient un enregistrement.

Après avoir permis à l'utilisateur d'entrer le contenu des 7 champs d'un enregistrement, nous devons lui permettre également de renouveler son entrée, au cas où il aurait commis une erreur.

```
15090 LOCATE 1,18
15100 PRINT"Donnees entrees correctement (o/n)";
15110 INPUT reponse$
15120 IF reponse$="o" THEN 15150
15130 IF reponse$="n" THEN compteur=compteur-1: GOSUB 300:GOTO
15000
15140 erreur$="Appuyez sur 'o' ou 'n' !":GOSUB 300: GOTO 15090
```

Si les données ont été entrées correctement, le programme continue, sinon la section d'entrée des données est à nouveau exécutée. Le compteur est cependant diminué auparavant de 1 pour annuler l'effet de la ligne 15030. Si la réponse fournie n'est pas correcte, l'ordinateur sort un message d'erreur et repose la question.

Le programme se poursuit avec à nouveau une question :

```
15150 LOCATE 1,20
15160 PRINT"D'autres entrees (o/n)";
15170 INPUT reponse$
15180 IF reponse$="o" THEN GOSUB 300:GOTO 15000
15190 IF reponse$="n" THEN 1000
15200 erreur$="Appuyez sur 'o' ou 'n' !":GOSUB 300: GOTO 15150
```

L'utilisateur doit décider ici s'il veut entrer ou non d'autres adresses. Si oui, cette partie du programme est relancée, si non, le programme retourne au menu. Une entrée incorrecte entraîne un message d'erreur et le renouvellement de la question.

Nous en avons maintenant terminé avec la première partie du programme. En voici une récapitulation :

```
15000 REM ====================
15010 REM  ENTRER LES ADRESSES
15020 REM ====================
15030 compteur=compteur+1
```

```
15040 PRINT
15050 FOR indice=1 TO 7
15060 PRINT champ$(indice);
15070 INPUT adresses$(compteur,indice)
15080 NEXT indice
15090 LOCATE 1,18
15100 PRINT"Donnees entrees correctement (o/n)";
15110 INPUT reponse$
15120 IF reponse$="o" THEN 15150
15130 IF reponse$="n" THEN compteur=compteur-1: GOSUB 300:GOTO
15000
15140 erreur$="Appuyez sur 'o' ou 'n' !":GOSUB 300: GOTO 15090
15150 LOCATE 1,20
15160 PRINT"D'autres entrees (o/n)";
15170 INPUT reponse$
15180 IF reponse$="o" THEN GOSUB 300:GOTO 15000
15190 IF reponse$="n" THEN 1000
15200 erreur$="Appuyez sur 'o' ou 'n' !":GOSUB 300: GOTO 15150
```

Modifier les adresses

Cette partie du programme doit permettre à l'utilisateur de modifier les différents champs d'un enregistrement déterminé. Nous utiliserons pour cela les touches du clavier. La touche 'v' permettra de "feuilleter" notre fichier en allant vers l'aVant et la touche 'r' en allant vers l'aRrière. Pour modifier une Adresse, l'utilisateur utilisera la touche 'a' et la touche [Return] permettra de mettre fin à la modification des données pour retourner au menu.

Facile à dire ! Maintenant à nous de traduire cela en Basic ! Commençons par le titre :

```
20000 REM ======================
20010 REM  MODIFIER LES ADRESSES
20020 REM ======================
```

Il nous faut maintenant une variable représentant le premier indice du tableau. Cette variable, 'pointeur' recevra comme valeur de départ 1:

```
20030 pointeur=1
```

Nous pouvons maintenant sortir la première adresse que contienne notre tableau. Pour sortir les numéros, noms et contenus des 7 champs de ce premier enregistrement, il nous suffit de réaliser une petite boucle :

```
20040 LOCATE 1,10
20050 FOR indice=1 TO 7
20060 PRINT indice;champ$(indice);adresses$ (pointeur,indice)
20070 NEXT indice
```

L'indice 'pointeur' donne le numéro d'enregistrement et l'indice 'indice' le numéro de champ.

Après que la première adresse ait été sortie, l'utilisateur doit choisir entre les 4 possibilités suivantes :

Touche 'v'	- feuilleter en aVant (adresse suivante)
Touche 'r'	- feuilleter en aRrière (dernière adresse)
Touche 'a'	- modifier l'Adresse montrée
[Return]	- retour au menu

Il nous faut maintenant interroger le clavier jusqu'à ce que l'utilisateur appuie sur une touche quelconque. Nous devons pour cela avoir recours à une nouvelle instruction, INKEY$ dont voici la description :

Problème :	Lecture d'un caractère entré au clavier	
Instruction :	chaîne=INKEY$	
Paramètres :	chaîne	variable alphanumérique dans laquelle sera placée la touche enfoncée

Exemple :	X$=INKEY$ Teste si une touche a été enfoncée et, si oui, place cette touche en X$.
Remarque :	Si aucune touche n'est enfoncée, le contenu de la variable alphanumérique sera une chaîne vide.

Si vous voulez que le programme ne continue que lorsqu'une touche aura été enfoncée, il faut entrer une ligne comme celle-ci (ce n'est qu'un exemple, ne l'entrez pas maintenant) :

```
10 X$=INKEY$:IF X$="" THEN 10
```

La ligne 10 sera donc exécutée en permanence, tant que l'utilisateur n'aura pas enfoncé une touche quelconque. Vous pouvez entrer maintenant la ligne suivante qui fera partie de notre programme :

```
20080 touche$=INKEY$:IF touche$="" THEN 20080
```

Lorsque l'utilisateur aura appuyé sur une touche, le programme passera à la ligne suivante. Il nous faut maintenant tester s'il a bien appuyé sur une des 4 touches v, r, a ou `Return`, et réagir en conséquence. Testons d'abord si l'utilisateur a appuyé sur la touche `Return` dont le code ASCII est 13 :

```
20090 IF touche$=CHR$(13) THEN 1000
```

Nous sautons donc au menu lorsque la touche `Return` est enfoncée. Si c'est la touche 'v' qui a été enfoncée, il faut sortir l'enregistrement suivant. Nous augmentons pour cela l'indice 'pointeur', sauf si 'pointeur' a déjà atteint le dernier enregistrement (pointeur=compteur). Voici donc la ligne qui remplit cette tâche :

```
20100 IF touche$="v" AND pointeur<compteur THEN
pointeur=pointeur+1:GOTO 20040
```

Ce n'est donc que si la touche 'v' a été enfoncée ET si la dernière adresse n'a pas encore été atteinte, que le pointeur sera fixé sur l'adresse suivante qui sera alors sortie.

Le principe est le même pour la touche 'r', sauf que c'est le contraire ! Le 'pointeur' doit donc être diminué de 1, sauf si 'pointeur' vaut déjà 1 :

```
20110 IF touche$="r" AND pointeur>1 THEN pointeur=pointeur-1
:GOTO 20040
```

Il nous faut encore tester si la touche 'a' a été enfoncée. Cette touche permet d'effectuer les modifications.

```
20120 IF touche$="a" THEN 20140
20130 erreur$="Entree incorrecte !":GOSUB 200: GOTO 20080
```

La ligne 20130 n'est atteinte que si l'utilisateur appuie sur les touches 'v' ou 'r' alors qu'il est déjà au bout du fichier, ou si l'utilisateur n'a appuyé sur aucune des 4 touches autorisées. Dans ce cas, le programme sort un message d'erreur et retourne à la ligne 20080 qui interroge à nouveau le clavier.

Il nous manque encore cependant la modification de l'enregistrement qui sera réalisée à partir de la ligne 20140. La modification consiste d'abord à demander le numéro puis le contenu du champ à modifier.

```
20140 LOCATE 1,18
20150 INPUT"Numero du champ(1-7): ";numero
20160 IF numero>0 AND numero<8 THEN 20180
20170 erreur$="Entrer un numero entre 1 et 7 !": GOSUB 200:GOTO
20140
20180 INPUT"Nouveau contenu: ";adresses$(pointeur, numero)
20190 GOSUB 300:GOTO 20040
```

Cette partie de programme est maintenant également terminée. Le numéro du champ à modifier est ici entré au clavier et placé dans la variable 'numero'. Puis nous testons si 'numero' a une valeur correcte. Si c'est le cas, la ligne 20180 permet d'entrer le nouveau contenu de ce champ. Le nouvel enregistrement est

alors sorti à partir de la ligne 20040. Si le numéro entré est incorrect, la ligne 20170 sort un message d'erreur. Cette partie du programme ne peut être abandonnée qu'en appuyant sur la touche [Return]. Voici maintenant une récapitulation de la partie "MODIFIER LES ADRESSES" :

```
20000 REM ======================
20010 REM  MODIFIER LES ADRESSES
20020 REM ======================
20030 pointeur=1
20040 LOCATE 1,10
20050 FOR indice=1 TO 7
20060 PRINT indice;champ$(indice);adresses$ (pointeur,indice)
20070 NEXT indice
20080 touche$=INKEY$:IF touche$="" THEN 20080
20090 IF touche$=CHR$(13) THEN 1000
20100 IF touche$="v" AND pointeur<compteur THEN pointeur
=pointeur+1:GOTO 20040
20110 IF touche$="r" AND pointeur>1 THEN pointeur=pointeur-1
:GOTO 20040
20120 IF touche$="a" THEN 20140
20130 erreur$="Entree incorrecte !":GOSUB 200: GOTO 20080
20140 LOCATE 1,18
20150 INPUT"Numero du champ(1-7): ";numero
20160 IF numero>0 AND numero<8 THEN 20180
20170 erreur$="Entrer un numero entre 1 et 7 !": GOSUB 200:GOTO
20140
20180 INPUT"Nouveau contenu: ";adresses$(pointeur, numero)
20190 GOSUB 300:GOTO 20040
```

Supprimer des adresses

Nous allons employer, pour la suppression des adresses, une autre technique que celle employée pour leur modification. Nous souhaitons en effet vous faire découvrir, à l'occasion du développement de ce programme, la plus grande variété de techniques de programmation possible.

L'utilisateur devra fournir le prénom et le nom correspondant à l'adresse à supprimer. Si ces données ne sont pas connues,

l'utilisateur pourra les déterminer en feuilletant les enregistrements avec la fonction "MODIFIER LES ADRESSES".

Cependant, avant même que ne commence cette partie du programme, encore faut-il contrôler qu'il y a bien des données en mémoire. Nous allons donc écrire en 500 une routine qui enverra un message d'erreur pour le cas où il n'y aurait pas de données en mémoire :

```
500 REM ====================
510 REM    Pas de donnees !
520 REM ====================
530 erreur$="Pas de donnees en memoire !"
540 GOSUB 200
550 RETURN
```

En fait le contrôle de la présence de données devrait être introduit également dans la partie "MODIFIER LES ADRESSES". Ajoutons donc la ligne suivante:

```
20025 IF compteur=0 THEN GOSUB 500:GOTO 1000
```

Cette ligne teste s'il y a des données en mémoire. La variable 'compteur' contient toujours le numéro du dernier enregistrement entré et donc aussi le nombre d'adresses stockées dans la mémoire de l'ordinateur. Si cette variable 'compteur' vaut 0, nous envoyons donc le message d'erreur 'Pas de données en mémoire !' avec la routine de la ligne 500, puis nous retournons au menu.

Nous pouvons maintenant commencer à écrire la nouvelle partie du programme :

```
25000 REM ======================
25010 REM  SUPPRIMER DES ADRESSES
25020 REM ======================
25030 IF compteur=0 THEN GOSUB 500:GOTO 1000
25040 LOCATE 1,10
25050 INPUT"Prenom: ";prenom$
25060 INPUT"Nom   : ";nom$
```

Après avoir contrôlé qu'il y a bien des données en mémoire, nous permettons l'entrée au clavier du prénom et du nom que nous plaçons dans les variables correspondantes. Il nous faut ensuite rechercher ces noms dans l'ensemble de notre tableau d'adresses. Nous ne pouvons pas utiliser ici une boucle FOR ... NEXT car nous aurions certainement à la quitter avant qu'elle ne soit terminée. Or chaque fois qu'on ouvre une boucle, l'adresse de début de la boucle est stockée par l'ordinateur qui ne l'efface qu'une fois la boucle terminée. Si l'on quitte donc plusieurs boucles avant qu'elles ne soient terminées, on gaspille de plus en plus de place en mémoire, ce qui peut même conduire à un arrêt du programme.

Construisons donc nous-même notre boucle :

```
25070 boucle=1
25080 IF adresses$(boucle,2)=prenom$ AND adresses$(boucle,3)=nom$
THEN 25110
25090 IF boucle<compteur THEN boucle=boucle+1:
GOTO 25080
25100 erreur$="Adresse inexistante !":GOSUB 200: GOTO 1000
```

La valeur initiale de la valeur boucle est d'abord fixée à 1. La boucle teste si l'adresse d'indice 'boucle' correspond à l'adresse recherchée. Le deuxième indice (2 et 3) correspond aux champs de prénom et de nom qui sont comparés à prenom$ et nom$. Si le nom correspond, le programme se poursuit à partir de la ligne 25110. La ligne 25090 augmente dans le cas contraire la variable de boucle de 1, sauf si la fin du tableau a déjà été atteinte.

Si la boucle a été parcourue jusqu'au bout, c'est que le nom recherché n'a pas été trouvé et un message d'erreur apparaît avant que le programme ne retourne au menu.

Il nous faut maintenant continuer l'écriture du programme à partir de la ligne 25110 pour traiter le cas où l'adresse recherchée a été trouvée. Dans ce cas, l'adresse à supprimer sera présentée entièrement pour que l'utilisateur puisse décider s'il convient bien de la supprimer.

```
25110 GOSUB 300:LOCATE 1,10
25120 FOR indice=1 TO 7
25130 PRINT champ$(indice);adresses$(boucle, indice)
25140 NEXT indice
25150 LOCATE 1,18
25160 INPUT"Supprimer cette adresse (o/n) "; reponse$
25170 IF reponse$="o" THEN 25200
25180 IF reponse$="n" THEN 1000
25190 erreur$="'Appuyer sur 'o' ou 'n' !":GOSUB 200: GOTO 25150
```

Nous construisons ici d'abord une nouvelle image d'écran puis nous positionnons le curseur (ligne 25110). La boucle des lignes 25120 à 25140 sort ensuite l'enregistrement complet de l'adresse. Nous demandons ensuite s'il faut bien supprimer cette adresse. Si non, le programme retourne au menu. Si l'adresse doit bien être supprimée, c'est ce que font les lignes 25200 et suivantes que voici :

```
25200 FOR i1=boucle TO compteur-1
25210 FOR i2=1 TO 7
25220 adresses$(i1,i2)=adresses$(i1+1,i2)
25230 NEXT i2
25240 NEXT i1
25250 compteur=compteur-1
25260 GOTO 1000
```

Il ne suffit pas, pour supprimer une adresse, d'annuler le contenu de l'emplacement du tableau correspondant. Dans ce cas en effet le tableau occupera toujours autant de place et l'ancien enregistrement ne pourra être remplacé effectivement par un autre enregistrement. Il faut donc décaler d'un emplacement vers le début du tableau tous les enregistrements situés après l'enregistrement supprimé. Si vous supprimez par exemple l'adresse d'indice 5 et qu'il y ait par exemple 7 adresses en mémoire, l'adresse 6 occupera ainsi la place de l'adresse 5 et la dernière adresse occupera la place de l'adresse 6. L'adresse d'indice 5 aura été ainsi véritablement supprimée.

Dans la programmation de ce principe, nous avons utilisé pour la première fois deux boucles imbriquées, puisque la boucle avec la variable i2 se trouve à l'intérieur de la boucle avec la

variable i1. La boucle i1 parcourt tous les enregistrements à partir de l'enregistrement à supprimer. Les champs de chaque enregistrement sont décalés un par un vers l'enregistrement précédent. Lorsque tous les champs d'un enregistrement ont été décalés, l'enregistrement suivant peut être traité. La boucle extérieure ne court que jusqu'à ce que i1 égale compteur-1 puisque la ligne 25220 utilise l'indice i1+1. Nous vous conseillons de bien étudier ces lignes qui vous présentent une technique importante en informatique.

Voici maintenant une récapitulation de cette partie du programme :

```
25000 REM ========================
25010 REM  SUPPRIMER DES ADRESSES
25020 REM ========================
25030 IF compteur=0 THEN GOSUB 500:GOTO 1000
25040 LOCATE 1,10
25050 INPUT"Prenom: ";prenom$
25060 INPUT"Nom   : ";nom$
25070 boucle=1
25080 IF adresses$(boucle,2)=prenom$ AND adresses$(boucle,3)=nom$
THEN 25110
25090 IF boucle<compteur THEN boucle=boucle+1:
GOTO 25080
25100 erreur$="Adresse inexistante !":GOSUB 200: GOTO 1000
25110 GOSUB 300:LOCATE 1,10
25120 FOR indice=1 TO 7
25130 PRINT champ$(indice);adresses$(boucle, indice)
25140 NEXT indice
25150 LOCATE 1,18
25160 INPUT"Supprimer cette adresse (o/n) "; reponse$
25170 IF reponse$="o" THEN 25200
25180 IF reponse$="n" THEN 1000
25190 erreur$="'Appuyer sur 'o' ou 'n' !":GOSUB 200: GOTO 25150
25200 FOR i1=boucle TO compteur-1
25210 FOR i2=1 TO 7
25220 adresses$(i1,i2)=adresses$(i1+1,i2)
25230 NEXT i2
25240 NEXT i1
```

```
25250 compteur=compteur-1
25260 GOTO 1000
```

Sortir les adresses

Cette partie n'est pas des plus faciles puisque nous allons devoir utiliser l'imprimante pour la première fois. Il faut également que l'utilisateur ait le choix de sortir les adresses qui l'intéressent à l'écran ou sur imprimante comme il le désire.

```
30000 REM ====================
30010 REM  SORTIR LES ADRESSES
30020 REM ====================
30030 IF compteur=0 THEN GOSUB 500:GOTO 1000
30040 LOCATE 1,10
30050 INPUT"Imprimante ou ecran (i/e) ";reponse$
30060 IF reponse$="i" THEN periph=8:GOTO 30090
30070 IF reponse$="e" THEN periph=0:GOTO 30090
30080 erreur$="'Appuyer sur 'i' ou 'e' !":GOSUB 200: GOTO 30040
```

Nous testons ici à nouveau s'il y a des données en mémoire (ligne 30030). Puis nous demandons si la sortie doit se faire sur écran ou imprimante. Nous affectons ensuite 0 ou 8 à la variable periph, en fonction du choix de l'utilisateur. Cette variable periph va nous servir pour une nouvelle instruction PRINT dont voici tout d'abord la description :

Problème :	Sortie sur différents périphériques	
Instruction :	PRINT #p, ...	
Paramètres :	p	numéro de périphérique (0-9) :
	0	l'écran
	1 à 7	une fenêtre de l'écran
	8	l'imprimante
	9	le lecteur de cassette
Exemple :	PRINT #8,"LISTE D'ADRESSES" Sort le texte "LISTE D'ADRESSES" sur imprimante.	

Remarque : - Le complément d'instruction '#p' peut être également utilisé en liaison avec les instructions LIST et INPUT que nous avons déjà rencontrées.
- PRINT #0 est équivalent à PRINT tout court.

L'instruction PRINT nous permet donc de sortir des données aussi bien sur imprimante ou sur cassette qu'à l'écran. Si nous utilisons donc l'instruction 'PRINT periph', la sortie se fera sur le périphérique choisi par l'utilisateur.

Il est également très intéressant de pouvoir préciser avec l'instruction LIST le périphérique de sortie. En effet LIST #8 vous permet par exemple de sortir un listing sur imprimante. Notez que le périphérique de sortie doit être indiqué après la zone de lignes à sortir :

LIST #8	- sort le programme complet sur imprimante
LIST 1000-,#8	- sort les lignes de programme, à partir de la ligne 1000, sur imprimante

Le complément d'instruction '#9' permet avec l'instruction INPUT d'entrer des données à partir de la disquette, mais nous y reviendrons plus tard. Revenons maintenant à notre programme.

Nous avons jusqu'ici déterminé le numéro du périphérique de sortie. Comme il n'est pas toujours nécessaire de sortir la totalité des adresses, nous allons fixer un critère de recherche pour chaque champ. Examinez les lignes suivantes :

```
30090 GOSUB 300:LOCATE 1,10
30100 PRINT"Criteres de recherche:"
30110 PRINT"----------------------"
30120 PRINT
30130 FOR indice=1 TO 7
30140 critere$(indice)=""
30150 PRINT champ$(indice);
30160 INPUT critere$(indice)
30170 NEXT indice
```

La ligne 30090 crée maintenant une nouvelle image d'écran puis les lignes 30100 - 30110 signalent qu'il faut maintenant entrer les critères de recherche. L'entrée se fait avec une autre boucle, après que les critères du tableau critere$ aient été annulés pour qu'ils correspondent bien aux critères pour cette recherche, et non aux critères résultant d'une recherche antérieure. Cette précaution n'a d'ailleurs d'intérêt que si vous utilisez ce programme sur un autre matériel que le CPC 6128 Plus qui annule en effet automatiquement le contenu d'une variable avant qu'un nouveau contenu ne soit entré avec INPUT. Les critères sont alors entrés en ligne 30160. Si vous voulez rechercher par exemple toutes les adresses de Paris, il vous suffit de n'entrer que [Return] pour les 4 premiers champs. Vous entrez alors pour le 5ème champ "Paris" puis [Return]. Vous pouvez alors ignorer également les 2 champs suivants avec [Return].

Une fois les critères de recherche entrés, la recherche peut commencer :

```
30180 FOR i1=1 TO compteur
30190 trouve=0
30200 FOR i2=1 TO 7
30210 IF critere$(i2)="" OR critere$(i2)=adresses$(i1,i2) THEN
trouve=trouve+1
30220 NEXT i2
```

Il nous faut à nouveau une boucle imbriquée qui examine tous les champs à l'intérieur d'un enregistrement. Au début de la boucle extérieure (ligne 30180) nous initialisons à 0 la variable 'trouve' qui comptera les champs qui correspondent au critère de recherche ou pour lesquels aucun critère de recherche n'a été entré. Si aucun critère de recherche n'a été entré pour un champ donné, la variable 'critere$(i1)' correspondante ne contient rien. La ligne 30120 examine donc les deux possibilités (absence de critère ou identité) et augmente la variable 'trouve' s'il y a lieu. Si, une fois que la boucle intérieure a été parcourue jusqu'au bout, le compteur d'identités 'trouve' vaut 7, c'est que l'adresse correspond aux critères de recherche indiqués et peut donc être sortie. Voici les lignes qui permettent cette sortie :

```
30230 IF trouve<>7 THEN 30300
30240 IF periph=0 THEN GOSUB 300
30250 PRINT #periph
```

Nous préparons ici la sortie de l'enregistrement trouvé. Si l'enregistrement ne correspond pas aux 7 critères de recherche, cet enregistrement ne sera pas sorti. Si c'est l'écran qui a été choisi, nous produisons un nouvel en-tête. Nous sortons ensuite une ligne vide sur le périphérique de sortie choisi. Voici maintenant les lignes correspondant au coeur de la sortie :

```
30260 FOR indice=1 TO 7
30270 PRINT #periph, champ$(indice);adresses$(i1,indice)
30280 NEXT indice
30290 PRINT #periph
30295 IF periph=0 THEN INPUT "Appuyez sur [Return] ";reponse$
30300 NEXT i1
30310 GOSUB 300:LOCATE 1,18
30320 PRINT"** FIN DU FICHIER **"
30330 INPUT"Appuyez sur [Return] ";reponse$
30340 GOTO 1000
```

Les lignes 30260 à 30280 construisent une boucle qui sortira les 7 champs d'une adresse avec leur désignation.

Si l'utilisateur a choisi la sortie sur écran, la recherche ne se poursuit qu'après que l'utilisateur ait appuyé sur [Return] (ligne 30295). Poursuivre la recherche signifie ici augmenter la variable de boucle 'i1' de 1 avec l'instruction 'NEXT i1'.

Une fois la boucle 'i1' terminée, la recherche est achevée, ce qui est indiqué par le message "** FIN DU FICHIER **". Si l'utilisateur appuie alors sur [Return], cette partie du programme est terminée et le programme retourne au menu.

Voici comme toujours la récapitulation de cette partie du programme :

```
30000 REM ====================
30010 REM  SORTIR LES ADRESSES
30020 REM ====================
```

```
30030 IF compteur=0 THEN GOSUB 500:GOTO 1000
30040 LOCATE 1,10
30050 INPUT"Imprimante ou ecran (i/e) ";reponse$
30060 IF reponse$="i" THEN periph=8:GOTO 30090
30070 IF reponse$="n" THEN periph=0:GOTO 30090
30080 erreur$="'Appuyer sur 'i' ou 'e' !":GOSUB 200: GOTO 30040
30090 GOSUB 300:LOCATE 1,10
30100 PRINT"Criteres de recherche:"
30110 PRINT"----------------------"
30120 PRINT
30130 FOR indice=1 TO 7
30140 critere$(indice)=""
30150 PRINT champ$(indice);
30160 INPUT critere$(indice)
30170 NEXT indice
30180 FOR i1=1 TO compteur
30190 trouve=0
30200 FOR i2=1 TO 7
30210 IF critere$(i2)="" OR critere$(i2)=adresses$(i1,i2) THEN
trouve=trouve+1
30220 NEXT i2
30230 IF trouve<>7 THEN 30300
30240 IF periph=0 THEN GOSUB 300
30250 PRINT #periph
30260 FOR indice=1 TO 7
30270 PRINT #periph, champ$(indice);adresses$(i1,indice)
30280 NEXT indice
30290 PRINT #periph
30295 IF periph=0 THEN INPUT "Appuyez sur [Return] ";reponse$
30300 NEXT i1
30310 GOSUB 300:LOCATE 1,18
30320 PRINT"** FIN DU FICHIER **"
30330 INPUT"Appuyez sur [Return] ";reponse$
30340 GOTO 1000
```

Sauvegarder le fichier

Nous approchons de la fin de notre programme de gestion d'adresses. Il nous faut maintenant sauvegarder les données entrées sur disquette, pour qu'elles ne soient pas perdues

chaque fois que vous éteindrez l'ordinateur. Voici déjà l'en-tête de cette nouvelle partie de programme :

```
10000 REM ========================
10020 REM  SAUVEGARDER LE FICHIER
10030 REM ========================
10040 IF compteur=0 THEN GOSUB 500:GOTO 1000
```

Il nous faut maintenant indiquer à l'utilisateur qu'il doit placer une disquette dans le lecteur de disquette :

```
10050 LOCATE 1,12
10060 PRINT"Veuillez introduire une disquette"
10070 PRINT
10080 INPUT"Appuyez ensuite sur [Return] ";reponse$
```

Jusque-là nous sommes en terrain connu. Mais maintenant, il nous faut préparer la disquette à recevoir des données. Nous devons donc "ouvrir" le fichier avec l'instruction dont voici la description :

Problème :	Ouverture d'un fichier en écriture	
Instruction :	OPENOUT "nom"	
Paramètres :	nom	nom du fichier
Exemple :	OPENOUT "ADRESSES" Ouvre le fichier "ADRESSES" en écriture	

Il faut donc ouvrir notre fichier avec cette instruction, avant de transmettre les données.

Comme nous utiliserons toujours le nom ADRESSES pour notre fichier, nous pouvons supprimer d'abord un éventuel fichier qui porterait déjà ce nom. Cela n'est normalement pas du tout nécessaire : en effet, lorsque le CPC 6128 Plus trouve sur la disquette un fichier portant le même nom que celui que vous voulez écrire, il transforme l'ancien fichier en fichier BACKUP. Il peut ainsi écrire ensuite le fichier sous le nom que vous avez

choisi. Cependant, dans le cas de fichiers de données de taille importante, le fichier BACKUP plus le nouveau fichier portant le même nom risquent d'occuper pas mal de place sur la disquette. C'est ce que nous évitons en supprimant d'abord tout fichier qui s'appellerait déjà ADRESSES. La suppression de l'ancien fichier et l'ouverture du fichier sont donc effectuées par les lignes suivantes :

```
10090 a$="Adresses":|ERA,@a$
10100 OPENOUT "Adresses"
```

Les données peuvent maintenant être sauvegardées. Nous écrirons d'abord sur la disquette le nombre d'enregistrements qui est contenu dans la variable 'compteur'. Puis nous enverrons les différents champs de notre tableau d'adresses à l'aide d'une boucle imbriquée, selon une technique que vous commencez à connaître : les champs 1 à 7 du premier enregistrement puis du second, etc... :

```
10110 PRINT #9,compteur
10120 FOR i1=1 TO compteur
10130 FOR i2=1 TO 7
10140 PRINT #9,adresses$(i1,i2)
10150 NEXT i2
10160 NEXT i1
```

Les données une fois sauvegardées, il faut refermer le fichier avec l'instruction 'CLOSEOUT' :

```
10170 CLOSEOUT
```

Nous pouvons ensuite indiquer à l'utilisateur que les données ont été sauvegardées :

```
10180 GOSUB 300
10190 PRINT"Les adresses ont ete sauvegardees !"
10200 FOR i=1 TO 2000:NEXT i
10210 GOTO 1000
```

Nous produisons ici d'abord une nouvelle image d'écran puis le message apparaît et reste à l'écran pendant un délai d'environ 2 secondes, après quoi le programme retourne au menu.

Vous voyez que la sauvegarde de données sur disquette est très simple. Il nous faut maintenant pouvoir récupérer les données ainsi sauvegardées.

Charger le fichier

Le chargement des adresses se déroule de façon pratiquement identique à la sauvegarde, sauf que nous ne contrôlerons pas s'il y a des données en mémoire. L'utilisateur devra donc penser à sauvegarder les données en mémoire, s'il y en a, avant de charger un autre fichier.

```
5000 REM ====================
5010 REM  CHARGER LE FICHIER
5020 REM ====================
5030 LOCATE 1,12
5040 PRINT"Veuillez introduire une disquette"
5050 PRINT
5060 INPUT"Appuyez ensuite sur [Return] ";reponse$
5070 PRINT
```

Après les préparatifs nécessaires, le fichier peut être ouvert, cette fois-ci en lecture. Voici la description de l'instruction qui convient ici :

Problème :	Ouverture d'un fichier en lecture	
Instruction :	OPENIN "nom"	
Paramètres :	nom	nom du fichier
Exemple :	OPENIN "ADRESSES" Ouvre le fichier "ADRESSES" en lecture	

Nous pouvons maintenant introduire cette instruction dans notre programme :

```
5080 OPENIN "Adresses"
5090 INPUT #9,compteur
5100 FOR i1=1 TO compteur
5110 FOR i2=1 TO 7
5120 INPUT #9,adresses$(i1,i2)
5130 NEXT i2
5140 NEXT i1
5150 CLOSEIN
```

Le fichier d'adresses est donc ici ouvert en lecture sur la disquette, puis le tableau d'adresses est reconstitué comme il avait été sauvegardé. Nous utilisons le nombre 'compteur' comme valeur finale de la boucle i1. On lira ainsi le même nombre d'adresses qu'on en a sauvegardé. Le canal de lecture est ensuite fermé en 5150.

```
5160 GOSUB 300:PRINT"Les adresses ont ete chargees !"
5170 FOR i=1 TO 2000:NEXT i
5180 GOTO 1000
```

Les dernières lignes de cette partie de programme sortent un message pour l'utilisateur puis retournent au menu après un délai.

Fin du programme

Comme nous l'avons déjà indiqué au début de ce chapitre, il n'est pas très élégant de devoir couper le courant pour arrêter un programme. Cette partie du programme permet donc d'obtenir une fin de programme plus convenable. En voici les premières lignes :

```
35000 REM =================
35010 REM  FIN DU PROGRAMME
35020 REM =================
35030 IF compteur=0 THEN 35150
```

Seule la ligne 35030 mérite un commentaire : si le compteur est à 0, cela veut dire qu'aucune adresse n'a encore été enregistrée et donc le programme peut être arrêté immédiatement sans problème, ce que fait la ligne 35150.

```
35040 LOCATE 1,12
35050 PRINT"Toutes les donnees ont ete sauvegardees (o/n) ";
35060 INPUT reponse$
35070 IF reponse$="n" THEN 1000
35080 IF reponse$="o" THEN 35100
35090 erreur$="Appuyer sur 'o' ou 'n' !":GOSUB 200: GOTO 35040
```

Cette partie de programme a donc pour objectif d'amener l'utilisateur à vérifier qu'il a bien sauvegardé toutes les données qu'il a pu entrer. S'il répond non, il retourne au menu. S'il est par contre sûr d'avoir sauvegardé les données, il peut entrer oui et le programme se termine ainsi :

```
35100 GOSUB 300:LOCATE 1,12
35110 PRINT"Le programme peut être relance avec GOTO 1000"
35120 PRINT"sans perte de donnees ! !"
35140 PRINT
35150 END
```

Le programme se termine donc par l'envoi à l'utilisateur d'une indication importante : le programme peut être relancé sans effacer les données avec l'instruction GOTO 1000. Par contre, si vous utilisez l'instruction RUN, toutes les variables seront détruites avant que le programme ne commence.

Nous sommes maintenant au bout de nos peines, en ce qui concerne ce programme de gestion d'adresses. Nous espérons que ce programme vous sera utile et que d'ores et déjà il vous a permis de vous familiariser avec certaines notions importantes.

6. Couleur et graphisme

Pour beaucoup d'acheteurs d'ordinateurs familiaux, les possibilités de couleurs et de graphisme constituent un critère de choix important. Le CPC possède, dans sa catégorie de prix, une splendeur de couleurs incomparable et d'excellentes caractéristiques graphiques. L'utilisateur d'un CPC 6128 Plus dispose ainsi de 32 couleurs parmi une palette de 4096 dont il peut utiliser simultanément, suivant la finesse de résolution graphique choisie, 2, 4 ou 16 d'entre elles. La résolution graphique atteint jusqu'à 640 points-écran sur 200, ce qui est proprement sensationnel pour un ordinateur de cette catégorie de prix.

Les modes graphiques

Le CPC 6128 Plus travaille avec 3 modes de résolution graphique différents qui peuvent être sélectionnés au moyen de l'instruction MODE que nous connaissons déjà bien. Il y a en effet un lien direct entre la densité d'écriture et la résolution graphique, comme le montre le tableau suivant :

Instruction	Caractères par lignes	Résolution graphique	Résolution Caractères	Couleurs
MODE 0	20	160 * 200	32 * 8	16
MODE 1	40	320 * 200	16 * 8	4
MODE 2	80	640 * 200	8 * 8	2

Un caractère en mode 2 se compose donc de 8 fois 8 points. La matrice d'un caractère passe à 16 sur 8 en mode 1, alors que la plupart des ordinateurs comparables représentent les caractères dans des matrices de 8 sur 8 même en 40 colonnes. Le CPC 6128 Plus dispose donc à cet égard d'une résolution graphique double de celle de la plupart des ordinateurs comparables.

En mode 2, chaque case mémoire stocke 8 points écran. La totalité de l'écran comporte 640 * 200, soit 128 000 points dont le

stockage requiert donc 16 000 cases mémoire. Sur les 128 kilo-octets dont dispose votre ordinateur, il en dépense pratiquement 16 pour le stockage du graphisme écran. La place mémoire ainsi utilisée pour gérer l'écran ne diminue pas si vous passez au mode 1 ou au mode 0 car la moindre résolution graphique est compensée par un plus grand nombre de couleurs disponibles et les informations nécessaires pour gérer les couleurs prennent la place laissée vacante par le nombre beaucoup plus réduit de points à gérer. Le CPC 6128 Plus dispose d'une seule mémoire écran qui est aussi la mémoire graphique, ce qui présente l'avantage, contrairement à ce qui se passe sur la plupart des autres ordinateurs, de permettre très simplement un mélange de textes et de graphismes.

Les couleurs de l'écran

Après avoir mis sous tension votre ordinateur, vous voyez apparaître les couleurs standard en MODE 1, bleu et jaune clair, à condition bien sûr que vous ayez un moniteur couleur. Voici maintenant un tableau de toutes les couleurs disponibles sur le CPC 6128 Plus :

Code couleur	Couleur	Code couleur	Couleur
0	noir	13	blanc
1	bleu	14	bleu pastel
2	bleu vif	15	orange
3	rouge	16	rose
4	magenta	17	magenta pastel
5	mauve	18	vert vif
6	rouge vif	19	vert marin
7	pourpre	20	turquoise vif
8	magenta vif	21	vert citron
9	vert	22	vert pastel
10	turquoise	23	turquoise pastel
11	bleu ciel	24	jaune vif
12	jaune	25	jaune pastel
		26	blanc brillant

L'utilisation concrète de ces codes vous en dira peut-être plus sur les couleurs qu'ils représentent que ces noms parfois un peu arbitraires.

Comme ce tableau l'indique, les couleurs standard en mode 1 sont donc le 1 (bleu) et le 24 (jaune vif ou jaune clair).

La couleur du cadre

Le cadre est cette partie de l'écran que vous ne pouvez pas atteindre avec le curseur. Vous pouvez également en fixer la couleur. Après mise sous tension de l'ordinateur, vous ne pouvez pas distinguer le cadre parce qu'il a la même couleur que le fond de l'écran. Mais voici maintenant la description de l'instruction vous permettant de fixer la couleur du cadre :

Problème :	Modification de la couleur du cadre	
Instruction :	BORDER couleur1, couleur2	
Paramètres :	couleur1 couleur2	code couleur n'est fourni que si la couleur du cadre de l'écran doit clignoter entre 'couleur1' et 'couleur2'
Exemple :	BORDER 0 Fixe le noir comme couleur du cadre BORDER 0,13 Fait clignoter le cadre entre noir et blanc	
Remarque :	Quel que soit le mode écran, vous pouvez utiliser chacune des 27 couleurs	

Vous pouvez par exemple faire que le cadre ait la même couleur que les caractères, en entrant :

```
BORDER 24
```

Vous voyez maintenant qu'il y a bien un cadre indépendant du fond de l'écran. Voici maintenant un petit programme qui vous permettra d'examiner toutes les couleurs que peut prendre le cadre :

```
10 FOR couleur=0 TO 26
20 BORDER couleur
30 X$=INKEY$:IF INKEY$="" THEN 30
40 NEXT couleur
```

Nous avons ici créé une petite boucle qui produit les différents codes couleur existants. Ces codes sont ensuite affectés au cadre en ligne 20. La ligne 30 attend ensuite que vous appuyiez sur une touche pour vous montrer la couleur suivante.

Comme l'indique la description de l'instruction BORDER, vous pouvez faire clignoter le cadre en permanence en fournissant deux codes séparés par une virgule. La couleur du cadre passera alors sans cesse du premier code au second et vice versa. Essayez par exemple l'instruction suivante qui choisit délibérément les couleurs offrant le plus grand contraste, noir et blanc brillant :

```
BORDER 0,26
```

Evidemment si vous tenez à votre vue, il vaut mieux ne pas travailler trop longtemps avec un cadre clignotant de cette façon. Supprimons donc le clignotement :

```
BORDER 1
```

Le clignotement peut être utilisé de façon spectaculaire dans les programmes pour souligner un message d'erreur.

Fixation de la vitesse de clignotement

Les concepteurs du CPC 6128 Plus ont vraiment pensé à tout puisqu'ils ont même prévu une instruction pour fixer la vitesse de clignotement des couleurs. En voici la description :

Problème :	Fixation de la vitesse de clignotement	
Instruction :	SPEED INK durée1, durée2	
Paramètres :	durée1 durée2	Durée de la première couleur Durée de la seconde couleur (en 50èmes de seconde)
Exemple :	SPEED INK 50,100 La première couleur dure une seconde et la deuxième deux secondes	
Remarque :	Les valeurs fournies pour durée1 et durée2 doivent être comprises entre 1 et 255, ce qui correspond respectivement à une durée allant de 0.02 secondes à 5.1 secondes	

Voici un programme illustrant l'emploi de cette instruction :

```
10 INPUT"Premiere couleur (0-26):";couleur1
20 INPUT"Duree (1-255):          ";duree1
30 INPUT"Seconde couleur (0-26):";couleur2
40 INPUT"Duree (1-255):          ";duree2
50 BORDER couleur1,couleur2
60 SPEED INK duree1,duree2
```

Vous pouvez donc entrer les deux couleurs et les durées correspondantes. Les deux dernières instructions feront alors clignoter le cadre.

Modification de la couleur des caractères

La couleur standard des caractères est jaune vif (code 24). Vous pouvez cependant modifier cette couleur avec l'instruction dont voici la description :

Problème :	Modification de la couleur des caractères	
Instruction :	PEN #n, registre	
Paramètres :	#n	numéro de la fenêtre écran (peut être ignoré)
	registre	numéro du registre de couleurs (0-15)
Exemple :	PEN 1 La couleur des caractères passe à la couleur du registre de couleurs 1	
Remarque :	Les registres de couleurs sont modifiés avec l'instruction INK	

Les registres sont mis en place par l'instruction INK. Ils contiennent les couleurs pouvant être effectivement employées à un moment donné. En effet, les 27 couleurs ne peuvent être utilisées simultanément mais seulement 2, 4 ou 16 d'entre elles, suivant le mode écran. Voici la description de l'instruction INK :

Problème :	Définition des couleurs disponibles	
Instruction :	INK registre,couleur1,couleur2	
Paramètres :	registre	numéro du registre de couleurs (0-15)
	couleur1	code couleur (0-26)
	couleur2	est fourni lorsqu'on veut un clignotement permanent entre couleur1 et couleur2
Exemple :	INK 1,3 Place la couleur rouge dans le registre de couleurs 1 INK 3,1,9 Fait clignoter la couleur du registre 3 entre bleu et vert	

Comme le plus grand nombre de couleurs disponibles simultanément est de 16 (en mode 0), il y a également 16 registres de couleurs dont les numéros vont de 0 à 15. Si vous

voulez donc faire passer au rouge la couleur des caractères, il faut d'abord placer la couleur 3 dans un registre de couleurs. Prenons par exemple le registre 1 :

```
INK 1,3
```

L'instruction PEN peut maintenant modifier la couleur des caractères en indiquant le registre 1 :

```
PEN 1
```

Les registres de couleurs contiennent déjà des couleurs lorsque vous allumez votre CPC 6128 Plus. Le tableau suivant indique de quelles couleurs il s'agit :

Registre de couleurs	Mode 0	Mode 1	Mode 2
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	1,24	20	1
15	16,11	6	24

Comme le choix de couleurs est limité en mode 1 et mode 2, les couleurs se répètent. Par exemple, en mode 1, les registres 4, 8 et 12 sont solidaires. Vous ne pouvez donc pas choisir plus de quatre couleurs en mode 1 et deux en mode 2.

L'instruction INK a aussi pour effet de modifier en même temps la couleur de tous les caractères qui ont été sortis auparavant sur l'écran avec la couleur de ce registre de couleurs. Un exemple : la couleur standard des caractères après allumage de l'ordinateur est tirée du registre 1. Si vous modifiez ensuite la couleur de ce registre, vous modifiez du même coup la couleur de tous les caractères qui avaient cette couleur. Entrez par exemple l'instruction suivante :

```
INK 1,13
```

Tous les caractères de l'écran deviennent immédiatement blancs. Si vous utilisez cependant la couleur du registre 2 pour écrire sur l'écran, une modification du registre 1 n'aura aucun effet.

Le registre de couleurs standard affecté au fond de l'écran est le registre 0. Donc si vous choisissez comme couleur des caractères la couleur du registre 0, les caractères auront la même couleur que le fond de l'écran et ne pourront donc plus être distingués. Faites-en l'expérience :

```
PEN 0
```

Le curseur disparaît. Pour le faire réapparaître, entrez à l'aveuglette l'instruction :

```
PEN 1
```

Le registre standard pour les caractères est donc le registre 1. Comme il faut toujours un registre spécifique pour le fond, il y a donc 3 différentes couleurs possibles pour les caractères en mode 1. Voici par exemple un programme qui sort trois lignes dans trois couleurs différentes :

```
10 MODE 1
20 INK 1,12:INK 2,9:INK 3,13
30 PEN 1:PRINT"jaune"
40 PEN 2:PRINT"vert"
50 PEN 3:PRINT"blanc"
```

Nous avons utilisé les registres 1 à 3 pour obtenir 3 couleurs d'écriture. La ligne 20 affecte les couleurs voulues aux trois registres et les lignes suivantes sortent trois lignes de couleur différente.

Vous pouvez également placer deux couleurs dans un registre de couleurs, ce qui provoque alors un clignotement. Faites un RESET ([Shift]-[Control]-[ESC]) puis modifiez la couleur des caractères en noir/blanc :

```
INK 1,0,13
```

Tous les caractères sur l'écran se mettent alors à clignoter, y compris les caractères que vous tapez au clavier. Si vous voulez pouvoir entrer des caractères dont la couleur ne dépende pas de celle des caractères déjà présents sur l'écran, entrez :

```
PEN 2
```

Vous pouvez aussi mettre fin au clignotement en entrant :

```
INK 1,24
```

Voici à nouveau un petit programme de démonstration :

```
10 MODE 1
20 INK 2,1,26
30 INK 3,26,1
40 PEN 2:PRINT "ICI ET LA"
50 PEN 3:PRINT "ICI ET LA"
60 PEN 1
```

Modification de la couleur du fond

Pour modifier également le fond, il nous faut faire connaissance avec une nouvelle instruction :

Problème :	Modification de la couleur du fond	
Instruction :	PAPER #n,registre	
Paramètres :	#n	numéro de la fenêtre de l'écran (peut être ignoré)
	registre	numéro du registre de couleurs (0-15)
Exemple :	PAPER 2 Affecte au fond la couleur du registre de couleurs 2	
Remarque :	Pour modifier la totalité de l'écran, l'instruction PAPER doit être suivie d'une instruction CLS	

Le registre standard affecté au fond de l'écran est le registre 0. Si vous lui affectez le registre 1, les caractères ne pourront plus être distingués puisque le registre 1 est le registre standard affecté aux caractères.

Utilisons maintenant l'instruction PAPER pour que le fond de l'écran apparaisse en rouge clair, couleur qui se trouve en principe dans le registre 3 :

```
PAPER 3
```

Vous voyez que seul le fond des caractères écrits après l'emploi de cette instruction a changé de couleur. Si vous voulez que tout le fond de l'écran change de couleur, il vous faut encore entrer l'instruction CLS pour vider l'écran.

Vous pouvez aussi faire clignoter le fond de l'écran si vous lui affectez un registre de couleurs qui a été défini avec deux couleurs. Nous allons par exemple faire clignoter le fond de l'écran entre blanc et noir. Faites donc un RESET ([Shift]-[Control]-[ESC]) et entrez les instructions suivantes :

```
INK 3,0,13
PAPER 3
CLS
```

Vous pouvez déterminer la fréquence du clignotement avec l'instruction SPEED INK. Pour que chaque couleur dure une seconde, entrez l'instruction suivante :

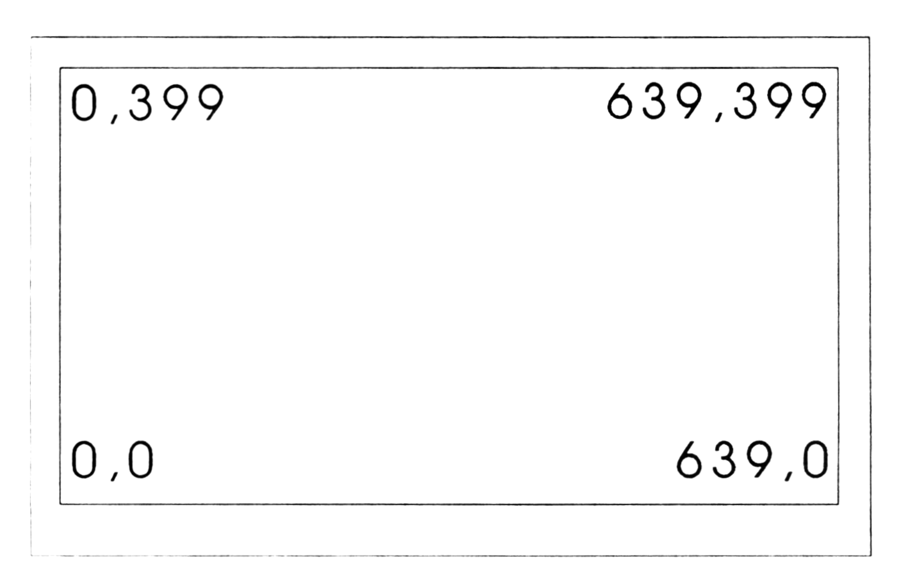
```
SPEED INK 50,50
```

Voici en conclusion un petit programme d'exemple spectaculaire :

```
10 MODE 1
20 INK 2,0,13
30 INK 3,13,0
40 SPEED INK 50,50
50 PEN 2
60 PAPER 3
70 CLS
80 PRINT "POSITIF ET NEGATIF"
```

Graphisme haute résolution

Le graphisme haute résolution vous permet de définir un écran de 640 points sur 200 en deux couleurs. Pour pouvoir commander chaque point de l'écran graphique, il faut se représenter l'écran comme un système de coordonnées. Chaque instruction graphique vous permet donc de définir un point en fournissant ses deux coordonnées, la première coordonnée qui correspond à l'axe des X en géométrie, c'est-à-dire à la position horizontale du point (entre le bord gauche et le bord droit de l'écran) et la seconde coordonnée qui correspond à l'axe des Y, c'est-à-dire à la position verticale du point (entre le bord supérieur et le bord inférieur de l'écran). Comme le rapport entre les deux dimensions de la haute résolution graphique (640 sur 200) ne correspond pas à la résolution effective de l'écran, les valeurs possibles des coordonnées Y ne vont pas de 0 à 199 mais, par paires (0 et 1, 2 et 3, etc...) de 0 à 399. Les quatre coins ont donc les coordonnées suivantes :



Si vous pensez avoir compris ce principe, nous pouvons passer à l'explication des instructions graphiques. Si vous n'avez pas compris, vous serez certainement éclairé par les exemples que nous donnons.

Dessin de points

Problème :	Allumer un point	
Instruction :	PLOT X,Y	
Paramètres :	X Y	position sur l'axe des X (0 - 639) position sur l'axe des Y (0 - 399)
Exemple :	PLOT 0,0 Place un point graphique dans l'angle inférieur gauche de l'écran	

Bien sûr, il est assez pénible de faire du graphisme, point par point, uniquement avec l'instruction PLOT et heureusement, votre CPC 6128 Plus dispose d'autres instructions très puissantes qui facilitent considérablement la réalisation de

graphiques. Mais pour que vous compreniez bien le principe des coordonnées, voici un programme d'exemple qui trace une ligne avec l'instruction PLOT employée dans une boucle. Cette ligne marquera le bord gauche de l'écran :

```
10 MODE 2
20 FOR Y=0 TO 399
30 PLOT 0,Y
40 NEXT Y
```

Compliquons un peu notre programme pour tracer maintenant un cadre tout autour de l'écran :

```
10 MODE 2
20 FOR Y=0 TO 399
30 PLOT 0,Y:PLOT 639,Y
40 NEXT Y
50 FOR X=0 TO 639
60 PLOT X,0:PLOT X,639
70 NEXT X
```

Nous avons ici usé d'un petit truc qui nous évite de créer quatre boucles pour les quatre côtés du cadre. En effet, il est très facile de tracer simultanément deux côtés parallèles. Mais nous pouvons encore améliorer ce petit programme, en ce qui concerne le tracé des lignes verticales. En effet, souvenez-vous que sur l'axe des Y, deux valeurs successives de Y désignent en fait le même point. Vous pouvez donc multiplier par deux la vitesse de création de la ligne en utilisant un incrément (ou un pas) de 2 dans la boucle pour les lignes verticales :

```
20 FOR Y=0 TO 399 STEP 2
```

Si vous lancez votre programme à nouveau après avoir effectué cette modification, vous pourrez constater l'accélération obtenue.

Dessin d'un cercle

Comme le CPC 6128 Plus ne dispose pas d'instruction spécifique pour dessiner les cercles, il faut allumer chaque point du cercle en appliquant la formule :

```
X = R * cos(a)
Y = R * sin(a)
```

R représente le rayon, et a l'angle. Il est ainsi possible de produire tous les points du cercle en créant une boucle qui trace les points correspondant aux 360 degrés de la circonférence :

```
10 MODE 2
20 X=320:Y=200:R=100
30 DEG
40 FOR a=1 TO 360
50 PLOT X+R*COS(a),Y+R*SIN(a)
60 NEXT a
```

La ligne 20 fixe la position du centre et le rayon du cercle. L'instruction DEG indique que tous les angles doivent être calculés en degrés (l'instruction contraire RAD a pour effet que tous les angles sont calculés en radians). La ligne 40 ouvre la boucle qui créera un cercle complet de 360 degrés. Si vous modifiez les valeurs initiale et finale, vous pouvez produire des portions de cercle. La formule de calcul se trouve en ligne 50. A la position du point ainsi calculée, il faut encore additionner le centre (X et Y).

Cette routine vous permet maintenant de tracer toutes sortes de cercle, même si vous ne comprenez rien à la géométrie. Si vous voulez réaliser des ellipses ou des figures encore plus complexes, il vous faut rechercher les formules mathématiques correspondantes et les traduire en Basic, comme nous l'avons fait pour le cercle.

Dessin d'une droite

Problème :	Dessin d'une droite
Instruction :	DRAW X,Y
Exemple :	DRAW 639,0 Dessine une droite entre la position du curseur graphique et la position X, Y fournie en paramètres

Le seul problème que pose cette instruction est le curseur. Lorsque vous allumez votre ordinateur, ce curseur se trouve en 0,0. Son emplacement est ensuite modifié par chaque instruction graphique. Si vous placez par exemple un point en 100,200, le curseur graphique se trouvera alors dans cet emplacement. Si vous utilisez alors l'instruction DRAW, c'est à partir de cet emplacement qu'une droite sera tracée. Le curseur graphique se trouvera ensuite sur le dernier point de cette droite.

Il est cependant possible de fixer l'emplacement du curseur graphique avec l'instruction suivante :

Problème :	Positionnement du curseur graphique
Instruction :	MOVE X,Y
Exemple :	MOVE 200,250 Place le curseur graphique dans l'emplacement 200,250

Nous pouvons maintenant dessiner beaucoup plus facilement une droite marquant le bord gauche de l'écran. Nous n'avons plus besoin que des deux instructions suivantes :

```
10 MODE 2
20 MOVE 0,0
30 DRAW 0,399
```

L'instruction MOVE 0,0 est inutile après l'allumage de l'ordinateur, tant que vous n'avez pas utilisé d'autre instruction graphique. Le curseur graphique se trouve en effet de toute façon en 0,0.

Nous pouvons maintenant, avec DRAW, tracer tout aussi aisément un cadre autour de l'écran :

```
10 MODE 2
20 MOVE 0,0:REM coin inferieur gauche
30 DRAW 0,399:REM vers le coin supérieur gauche
40 DRAW 639,399:REM vers le coin supérieur droit
50 DRAW 639,0:REM vers le coin inférieur droit
60 DRAW 0,0:REM vers le coin inférieur gauche
```

Vous voyez que la programmation des lignes devient très facile. L'instruction DRAW utilisée dans une boucle permet d'autre part d'obtenir des effets très intéressants :

```
10 MODE 2
20 FOR i=0 TO 399 STEP 10
30 MOVE 399,i
40 DRAW 0,399-i
50 NEXT i
```

Vous voyez qu'on peut obtenir des effets spectaculaires avec un tout petit programme. En voici un autre exemple :

```
10 MODE 2
20 MOVE 0,0
30 FOR i=0 TO 399 STEP 10
40 DRAW 399,i
50 DRAW 399-i,399
60 DRAW 0,399-i
70 DRAW i,0
80 NEXT i
```

Surprenant, non ? Amusez-vous aussi à réaliser des graphiques en laissant aller votre imagination. La seule difficulté est de traduire les figures que vous imaginez dans le système de coordonnées. Vous ne réussirez pas toujours du premier coup,

mais c'est comme cela que vous apprendrez le plus rapidement à maîtriser les possibilités graphiques de votre ordinateur.

Dessin relatif

En conclusion de ce chapitre, nous allons maintenant vous présenter une variante des instructions graphiques que nous vous avons présentées jusqu'ici. Les instructions PLOT, DRAW et MOVE possèdent en effet des soeurs comme le montrent les lignes ci-dessous :

```
PLOTR x,y
DRAWR x,y
MOVER x,y
```

Ces instructions se présentent avec deux coordonnées, comme celles que nous connaissons déjà. Cependant les coordonnées x,y ne sont plus ici des coordonnées absolues, ce qui signifie que les mêmes coordonnées ne correspondront pas toujours obligatoirement au même point. Il s'agit en effet de coordonnées relatives qui doivent être ajoutées aux coordonnées actuelles du curseur graphique pour calculer les coordonnées effectives du point concerné par l'instruction.

Par exemple, l'instruction 'PLOTR 50,50' aura pour effet d'allumer le point de coordonnées 250,150 si le curseur graphique était auparavant en 200,100 mais le point 60,200 si le curseur graphique était en 10,150. L'intérêt de ces instructions est donc de permettre d'indiquer le déplacement que doit effectuer le curseur graphique.

L'exemple suivant qui trace un carré de 10 points sur 10 va vous permettre d'y voir plus clair. Les coordonnées de l'angle supérieur gauche de ce carré se trouvent dans les variables x et y :

```
10 MODE 2
20 x=200,y=100
30 MOVE x,y
40 DRAWR 10,0:REM 10 vers la droite
```

```
50 DRAWR 0,10:REM 10 vers le haut
60 DRAWR -10,0:REM 10 vers la gauche
70 DRAWR 0,-10:REM 10 vers le bas
```

Bien sûr, comme vous le voyez dans notre exemple, les coordonnées doivent avoir des valeurs négatives lorsque vous voulez déplacer le curseur vers la gauche ou vers le bas.

7. La production de sons

Un ordinateur se prête parfaitement à la production de sons. Le processeur, qui est le coeur de l'ordinateur, est en effet commandé par un générateur de fréquences d'environ 1 à 4 méga Hertz, suivant les ordinateurs. Il serait donc très simple de diffuser ces fréquences avec un haut-parleur, mais l'oreille ne peut entendre des fréquences aussi élevées. Pour réaliser de la musique électronique, on divise donc, avant la sortie sur haut-parleur, les fréquences produites, de façon à obtenir des sons perçus comme différentes notes.

C'est ainsi qu'ont été produits les premiers sons électroniques. Puis sont apparus les synthétiseurs qui permettent d'obtenir des sons ressemblant étrangement aux sons naturels des instruments de musique. Le chip sonore du CPC 6128 Plus n'est pas un synthétiseur à proprement parler car il produit les fréquences par subdivision.

Le générateur de sons du CPC 6128 Plus dispose de trois voix qui vous permettent donc de produire trois notes différentes simultanément. Le CPC 6128 Plus dispose également d'un générateur de bruits.

Les sons produits par le générateur de sons sortent par le haut-parleur intégré de votre CPC. Il est donc préférable de couper le son de votre téléviseur si vous l'utilisez comme moniteur de votre ordinateur. Vous pouvez régler le volume du haut-parleur intégré avec le bouton placé sous la face avant du moniteur.

Voici maintenant une introduction aux possibilités sonores de votre ordinateur. Nous vous présentons ici les diverses instructions sonores dont dispose le CPC 6128 Plus.

L'instruction Sound

Problème :	Définition des caractéristiques d'une note	
Instruction :	SOUND a,b,c,d,e,f,g	
Paramètres :	a	Canal d'état (toujours 1 dans ce chapitre)
	b	Diviseur, définit la fréquence (0 - 4095) Fréquence = 125000/diviseur
	c	Durée de la note (-32768 à 32767 en centièmes de seconde)
	d	Volume (0-15)
	e	Courbe d'enveloppe du volume (0-15)
	f	Courbe d'enveloppe de la note (0-15)
	g	Définition d'un bruit (0-31)
Exemple :	SOUND 1,1000,100,7,0,0,0 Produit une note de 125 Herz d'une durée de 1 seconde avec un volume de 7	

Nous ne pouvons décrire ici tous les paramètres de cette instruction dans le détail. Nous voulons simplement vous mettre le pied à l'étrier pour que vous puissiez ensuite approfondir vous-même cette question avec votre manuel.

La hauteur de note

Dans ce chapitre, nous utiliserons toujours 1 pour le premier paramètre qui correspond au canal d'état et qui permet notamment un jeu simultané des trois voix.

Le second paramètre, b, détermine la hauteur de la note. Il s'agit d'un diviseur de la fréquence de base, 125 000 Herz. Comme la valeur de ce paramètre peut varier de 0 à 4095, vous pouvez produire des fréquences entre 30 et 125000 Herz. Le programme suivant vous permet d'écouter des fréquences de 30000 à 300 Herz :

```
10 FOR diviseur=4.1 TO 416 STEP 0.1
20 PRINT INT(125000/diviseur)
30 SOUND 1,diviseur,1,7,0,0,0
40 NEXT diviseur
```

Vous voyez que dans l'instruction SOUND, comme dans toutes les instructions Basic, les paramètres peuvent tous être représentés par des variables.

L'utilisation d'une boucle pour modifier la hauteur d'une note autorise toutes sortes d'effets. Voici par exemple comment simuler une sirène :

```
10 FOR diviseur=120 TO 200
20 SOUND 1,diviseur,2,7,0,0,0
30 NEXT diviseur
40 FOR diviseur=200 TO 120 STEP -1
50 SOUND 1,diviseur,2,7,0,0,0
60 NEXT diviseur
70 GOTO 10
```

Bien sûr, il est souhaitable de pouvoir arrêter cette sirène avant qu'elle ne vous rende fou. Il faut pour cela appuyer deux fois sur la touche [ESC]. Mais si vous succombez volontiers au charme des sirènes, voici une variante du programme précédent :

```
5 INPUT "Diviseur initial";a
6 INPUT "Diviseur final";b
7 INPUT "Tempo (1-10)";t
10 FOR diviseur=a TO b
20 SOUND 1,diviseur,t,7,0,0,0
30 NEXT diviseur
40 FOR diviseur=b TO a STEP -1
50 SOUND 1,diviseur,t,7,0,0,0
60 NEXT diviseur
70 GOTO 10
```

Faire de la musique en partant des fréquences

Les instruments de musique ne produisent pas des suites continues de fréquences. La musique repose en effet sur les intervalles existant entre les notes et l'oreille humaine ne perçoit pas les intervalles trop réduits. La musique en général repose donc sur une échelle de fréquences et la musique occidentale repose sur une échelle de 12 tons qui constituent la gamme diatonique que vous obtenez en appuyant successivement sur 12 touches consécutives d'un piano, sans oublier les touches noires :

```
TOUCHES      DO   RE        FA   SOL  LA
NOIRES        #    #         #    #    #
TOUCHES    DO   RE   MI   FA   SOL  LA   SI
BLANCHES
POSITION
DU TON     1 2  3 4  5    6 7  8  9 10 11 12
```

Cette suite de tons se répète plusieurs fois sur un piano. Chaque groupe de 12 tons successifs constitue une octave. La première octave contient les sons les plus graves, la dernière les sons les plus aigus.

Comment obtenir maintenant de votre ordinateur la production de tons correspondant à ceux des instruments de musique ? Sachez d'abord que le LA de l'octave située sur la clé du piano a pour fréquence internationalement reconnue 440 Herz. Pour produire cette note, divisez 125000 par 440, vous obtenez 284.090909. Vous pouvez donc produire le LA du diapason avec l'instruction :

```
SOUND 1,284,100,7,0,0,0
```

Pour calculer maintenant tous les tons de la gamme diatonique, il faut utiliser la formule suivante :

```
FREQUENCE = 440 * (2^(OCTAVE+(N-10)/12))
```

OCTAVE peut varier de -3 à 4

N correspond à la position de la note sur la gamme diatonique, telle que nous vous l'avons fournie plus haut (Do=1, La=10).

Peu importe l'origine mathématique de cette formule, ce qui compte, c'est que vous soyez à même de l'utiliser. Si vous voulez produire par exemple le DO de l'octave la plus basse, vous pouvez en calculer la fréquence ainsi :

```
440 * (2^(-3+(1-10)/12))=32.7031957 Herz
```

Pour produire la fréquence ainsi calculée, il nous faut maintenant calculer le diviseur :

```
125000/32.7031957 = 3822.25643
```

Le diviseur 3822 produira donc le DO de l'octave la plus basse. En suivant le même principe, nous pouvons écrire un programme jouant la gamme diatonique :

```
10 INPUT"Octave (-3 a 4)";octave
20 FOR ton=1 TO 12
30 frequence=440*(2^(octave+(ton-10)/12))
40 diviseur=INT(125000/frequence)
50 SOUND 1,diviseur,20,7,0,0,0
60 NEXT ton
```

Et voici maintenant le clou de ce chapitre, un programme qui transformera votre CPC en un instrument de musique :

```
10 MODE 0
20 DIM ton(90)
30 ton$="A2Z3ER5T6Y7UI9O0P"
40 FOR i=1 TO 17
50 indice=ASC(MID$(ton$,i,1))
60 ton(indice)=i
70 NEXT i
80 CLS
90 PRINT"LE PIANO POUR DEBUTANTS"
100 PRINT
110 LOCATE 1,5
120 PRINT" 2 3   5 6 7   9 0"
```

```
130 PRINT
140 PRINT"A Z E R T Y U I O P"
150 a$=INKEY$:IF a$="" THEN 150
160 IF ton(asc(a$))=0 THEN 150
170 ton=ton(asc(a$))
180 frequence=440 *(2^(ton/12))
190 diviseur=INT(125000/frequence)
200 SOUND 1,diviseur,30,7,0,0,0
210 GOTO 150
```

Avant de lancer ce programme, appuyez sur les touches [Control] et [CapsLock] en même temps.

Les bruits

Délaissons maintenant les hauteurs de l'harmonie musicale pour en venir aux bruits. Vous avez vu, dans la description de l'instruction SOUND, que le dernier paramètre de cette instruction permet de produire des bruits. Ce paramètre doit pour cela recevoir une valeur comprise entre 1 et 31. L'instruction SOUND permet donc de produire aussi bien des notes que des bruits, suivant les valeurs des paramètres 'b' et 'g', ainsi que l'explique le tableau suivant :

SOUND a,b,c,d,e,f,g			
Paramètre		Résultat	
'b'	'g'	NOTE	BRUIT
0	0	non	non
1-4095	0	oui	non
0	1-31	non	oui
1-4095	1-31	oui	oui

Vous voyez que l'instruction SOUND permet d'émettre simultanément des sons et des bruits.

Le programme suivant montre les nuances de bruit que vous pouvez obtenir suivant la valeur du dernier paramètre :

```
10 FOR r=1 TO 31
20 SOUND 1,0,20,7,0,0,r
30 NEXT r
```

On entend nettement les variations du bruit. La valeur 0 du second paramètre déconnecte la production de tons musicaux.

La production de bruits permet d'obtenir toutes sortes d'effets. Voici par exemple un programme imitant le bruit d'une locomotive :

```
10 SOUND 1,0,10,7,0,0,15
20 SOUND 1,0,5,7,0,0,0
30 GOTO 10
```

Vous pouvez arrêter cette locomotive en tirant le signal d'alarme ou plutôt la touche [ESC]. Ici le bruit dure un dixième de seconde puis il est suivi d'une pause d'un vingtième de seconde. Rappelez-vous que la durée d'un son ou d'un bruit est fixée par le troisième paramètre en centièmes de seconde.

Voici maintenant un dernier exemple qui imite une détonation, grâce à une boucle qui diminue le volume :

```
10 a$=INKEY$:IF a$="" THEN 10
20 FOR v=7 TO 1 STEP -1
30 SOUND 1,0,10,v,0,0,15
40 NEXT v
50 GOTO 10
```

Les coups partent lorsque vous appuyez sur une touche. Vous pouvez interrompre le programme avec [ESC]. Maintenant que nous avons décrit et expérimenté l'instruction SOUND, nous espérons que vous vous essaierez vous-même au bruitage électronique.

8. Le lecteur de disquette

Vous savez déjà à quoi sert le lecteur de disquette intégré de votre ordinateur puisque nous vous avons expliqué comment sauvegarder puis recharger un programme et que nous avons également décrit, à l'occasion du programme de gestion d'adresses, comment sauvegarder ou charger des données sur disquette. Ce chapitre explique donc les autres possibilités du lecteur de disquette.

Les disquettes

Les lecteurs de disquette fournis par AMSTRAD diffèrent des lecteurs ordinaires sur un point essentiel, la taille des disquettes. Après que les disquettes 5 pouces 1/4 aient été utilisées et soient encore utilisées depuis de nombreuses années, les disquettes de 3 pouces 1/2 semblent pouvoir amener sur le marché de l'informatique un nouveau standard. Les ordinateurs de la dernière génération (AMIGA, 520ST, MacIntosh) sont dotés de ce format. Les lecteurs de disquette AMSTRAD, par contre, ont un format qui ne figure, pour le moment, sur aucune autre machine : le format 3 pouces.

Les disquettes du CPC présentent un avantage par rapport aux disquettes 3 pouces 1/2; elles peuvent en effet être écrites sur deux faces, puisqu'il est possible de les retourner. Vous avez ainsi sur une disquette une capacité de deux fois 180 kilo-octets. La disquette est organisée physiquement en 40 pistes. Chacune des ces pistes est, à son tour, divisée en 9 secteurs. Sur chacun de ces 360 secteurs, il est possible de stocker 512 octets. Cela donne donc une capacité totale par face de 180 kilo-octets. Mais rassurez-vous, cette structure physique est sans importance pour vous. Le système d'exploitation ou, plus précisément, le DOS (Disk Operating System) se charge lui-même de tout le travail et organise la structure de la disquette.

Le formatage

Avant que vous ne puissiez utiliser une disquette pour y stocker des données, vous devez la formater. Le formatage consiste à organiser les secteurs sur la disquette, de façon à ce qu'ils puissent recevoir des données. Voici tout d'abord une description précise de la procédure à suivre pour formater une disquette :

1. Placez la disquette système CP/M (fournie avec la machine) dans le lecteur de disquette, face A dessus.

2. Appelez le système d'exploitation CP/M avec l'instruction ùCPM

3. Chargez le programme de formatage d'une disquette en entrant DISCKIT3.

4. Appuyez sur la touche de fonction f4 pour sélectionner le formatage.

5. Appuyez sur la touche de fonction f6 pour sélectionner le format de données.

6. Introduisez la disquette à formater dans le lecteur de disquette.

7. Appuyez sur la touche Y.

8. Une fois le formatage achevé, appuyez sur une touche quelconque.

9. Replacez la disquette système dans le lecteur de disquette.

10. Sortez du programme utilitaire en appuyant la touche de fonction f0.

11. Ré-initialisez votre ordinateur ([Shift]-[Control]-[ESC])

Il s'agit d'une procédure un peu compliquée, mais incontournable. Une fois qu'elle aura été exécutée avec succès,

vous disposerez d'une disquette formatée que vous pourrez utiliser pour le stockage de données.

Noms de fichier

Lorsque vous choisissez un nom pour vos programmes ou fichiers, vous devez tenir compte des points suivants :

- Les noms de fichier ne doivent pas comprendre plus de 8 caractères.
- Vous disposez de trois caractères supplémentaires, si vous séparez le nom de fichier de ces trois caractères par un point.

Les programmes BASIC sauvegardés normalement reçoivent automatiquement le suffixe .BAS si vous ne leur affectez pas vous-même un autre suffixe. Exemple :

SAVE"TEST"	Le programme sera sauvegardé sur la disquette sous le nom "TEST.BAS".
SAVE"TEST.DEM"	Le programme sera sauvegardé sur la disquette sous le nom "TEST.DEM".

Le catalogue de la disquette

Entrez quelques lignes BASIC et sauvegardez le programme ainsi créé sous le nom TESTCAT.

```
SAVE"TESTCAT"
```

Le catalogue de la disquette est sorti sur l'écran, lorsque vous en avez besoin, avec l'instruction CAT.

```
CAT
```

Si vous entrez maintenant cette instruction, vous voyez apparaître le catalogue suivant :

Drive A: user 0
TESTCAT .BAS 1K
177K free

Vous voyez que votre programme n'a pas une taille de plus d'un kilo-octets et que vous disposez encore d'une place de 177 Koctets sur cette disquette.

Pour charger le programme, entrez l'instruction suivante :

```
LOAD"TESTCAT"
```

ou

```
LOAD"TESTCAT.BAS"
```

Nous allons décrire d'autres possibilités pour la gestion de programmes.

Les programmes sous forme de fichiers ASCII

Nous avons souvent utilisé dans cet ouvrage les termes de fichier et de programme en veillant toujours à bien distinguer ces deux notions. Mais où est la différence, puisque les fichiers comme les programmes peuvent être stockés dans la mémoire de l'ordinateur ou être sauvegardés sur disquette ? La différence est que les programmes ne figurent pas, ni dans la mémoire de l'ordinateur, ni sur disquette, sous la forme sous laquelle ils apparaissent lorsque vous les listez. En effet les instructions sont remplacées dans les programmes par des abréviations. C'est ainsi que l'instruction PRINT d'un programme est représentée, comme toutes les instructions Basic, par un code (token), ce qui fait qu'elle n'occupe qu'une case mémoire et non pas cinq comme ce serait le cas si elle était stockée lettre par lettre.

Le CPC vous permet cependant de sauvegarder vos programmes sur disquette "en clair", c'est-à-dire sans coder les instructions. Dans ce cas, chaque lettre du programme est

sauvegardée sous la forme de son code ASCII. Voici la description de l'instruction qui permet cela :

Problème :	Sauvegarde de programmes en clair
Instruction :	SAVE "nom",A
Paramètres :	"nom" nom du programme
Exemple :	SAVE "Adresses",A Sauvegarde en clair le programme en mémoire sous le nom "Adresses"
Remarque :	Le programme ainsi sauvegardé peut être lu comme un fichier, et peut tourner comme un programme

L'intérêt de la sauvegarde des programmes sous cette forme est que vous pouvez alors les utiliser en traitement de texte et les insérer normalement dans un texte, comme nous le faisons dans le présent ouvrage.

L'exemple suivant vous montre comment ces programmes peuvent ensuite être lus comme des fichiers. Entrez d'abord le programme suivant :

```
10 REM ================================
20 REM  PROGRAMME SOUS FORME DE FICHIER
30 REM ================================
40 PRINT "LE PROGRAMME EST TERMINE"
```

Sauvegardez maintenant ce programme avec l'instruction suivante :

```
SAVE "DEMO.ASC",A
```

Vous pouvez maintenant lire ce programme et le sortir sur écran avec le programme suivant :

```
10 OPENIN "DEMO.ASC"
20 FOR i=1 TO 4
```

```
30 INPUT #9,a$
40 PRINT a$
50 NEXT i
60 CLOSEOUT
```

Programmes protégés

Si vous avez déjà essayé de charger des programmes commerciaux dans votre CPC puis de les lister, vous avez sans doute remarqué que c'était impossible. C'est que ces programmes étaient protégés. Vous pouvez protéger vous-même les programmes que vous réalisez en utilisant l'instruction dont voici la description :

Problème :	Protection de programmes sur disquette	
Instruction :	SAVE "nom",P	
Paramètres :	"nom"	nom du programme
Exemple :	SAVE "SECRET",P Sauvegarde le programme en mémoire sous le nom "SECRET", de telle façon qu'il ne pourra plus être listé	
Remarque :	Il faut absolument que vous sauvegardiez également une version non protégée de vos programmes protégés, sinon vous ne pourrez plus y travailler	

Voici maintenant comment vous pouvez tester immédiatement l'efficacité de cette instruction ; entrez le programme suivant :

```
10 REM ==================
20 REM  PROGRAMME SECRET
30 REM ==================
40 PRINT "Entrez le code:";
50 INPUT a$
60 IF a$<>"xy12" THEN NEW
70 PRINT "o.k."
```

Sauvegardez maintenant ce programme avec :

```
SAVE "SECRET.P",P
```

Chargez maintenant le programme avec :

```
LOAD "SECRET.P"
```

Lancez maintenant le programme avec 'RUN'. Rien ne se passe car les programmes protégés ne peuvent pas être chargés et lancés avec LOAD et RUN. Vous ne pouvez les charger et les lancer qu'avec l'instruction RUN"nom". Entrez donc maintenant l'instruction correcte :

```
RUN "SECRET.P"
```

Le programme sera alors chargé puis lancé automatiquement. Vous croyez peut-être que vous pouvez encore décoder ce programme ? Alors essayez de l'interrompre en appuyant deux fois sur la touche [ESC]. Vous voyez, pas de problème, vous avez bien interrompu le programme. Mais ne vous réjouissez pas trop vite car si vous entrez maintenant l'instruction LIST, vous verrez que rien ne se passera car le programme a préféré se faire hara-kiri plutôt que de livrer ses secrets à l'ennemi. Le programme a en effet été détruit au moment où vous l'avez interrompu avec la touche [ESC]. Bien entendu, cela ne signifie pas que ce code soit inviolable. Mais il faut pour cela de bonnes connaissances en langage-machine. Résumons-nous :

➤ ',P' placé à la suite de l'instruction SAVE normale protège un programme contre le listage

➤ Le programme ne peut plus être alors chargé et lancé qu'avec l'instruction RUN"nom"

➤ La protection est définitive

➤ Si le programme est interrompu avec [ESC], la mémoire est vidée de son contenu

Sauvegarde de zones de la mémoire

La mémoire du CPC comprend 2 fois 64 kilo-octets, soit 2 fois 65536 octets. Si vous voulez sauvegarder certaines parties de cette mémoire, vous pouvez utiliser l'instruction suivante :

Problème :	Sauvegarde de zones de la mémoire	
Instruction :	SAVE "nom",B,début,longueur,exécution	
Paramètres :	"nom"	nom du programme
	début	adresse de début de la zone (0 - 65535)
	longueur	longueur de la zone
	exécution	adresse d'exécution du programme (pour les programmes en langage-machine)
Exemple :	SAVE "routine.BIN",B,49152,16384 Sauvegarde tout le contenu de l'écran	
Remarque :	La zone de la mémoire ainsi sauvegardée doit être chargée avec l'instruction LOAD".	

Cette instruction est essentiellement destinée aux programmeurs expérimentés qui travaillent en langage-machine.

Elle peut cependant également être utilisée pour sauvegarder des graphiques ou dessins. Il faut pour cela sauvegarder la totalité de l'écran (49152 à 49152+16384) sur une disquette. Il peut être intéressant de sauvegarder ainsi des graphiques particulièrement compliqués dont la réalisation par un programme Basic prend plus de temps que le chargement d'un écran complet. C'est notamment le cas pour des fonctions à trois dimensions.

Fusion de programmes

Chaque programmeur développe avec le temps des sous-programmes ou routines particulièrement utiles qui peuvent être employés dans différents programmes. Une riche bibliothèque de sous-programme facilite considérablement le développement de nouveaux programmes. Pour fusionner un sous-programme avec le programme principal, utilisez l'instruction :

Problème :	Fusion de programmes
Instruction :	MERGE "nom"
Paramètres :	"nom" nom du programme
Exemple :	MERGE "SP01" Charge le programme "SP01" à la suite du programme se trouvant en mémoire
Remarque :	Si le programme chargé à la suite du programme en mémoire contient des numéros de ligne identiques à certains numéros de ligne du programme en mémoire, ce sont les lignes du programme en mémoire qui seront effacées

Le tableau suivant explique le fonctionnement de cette instruction :

Progr. en mémoire	Progr. sur disquette	Résultat
10 REM P1 20 REM P1	5 REM P2 15 REM P2	5 REM P2 10 REM P1 15 REM P2 20 REM P1
10 REM P1 20 REM P1	5 REM P2 10 REM P2	5 REM P2 10 REM P2 20 REM P1

Vous voyez que les lignes des deux programmes sont triées et ordonnées d'aprés les numéros de ligne. Mais si deux numéros de ligne sont identiques, c'est la ligne portant ce numéro dans le programme en mémoire (ici 10 REM P1) qui est détruite, de façon à ce qu'il n'y ait jamais deux lignes portant le même numéro.

Notez qu'il est très facile d'éviter ce genre de problèmes avec l'instruction RENUM que nous avons déjà expliquée et qui vous permet de renuméroter les lignes du programme en mémoire, de façon à ce qu'elles ne se chevauchent pas avec celles du sous-programme à fusionner.

Voici maintenant un exemple d'utilisation de l'instruction MERGE. Entrez d'abord le programme suivant :

```
10 REM programme 2
25 REM programme 2
30 REM programme 2
45 REM programme 2
```

Sauvegardez-le maintenant sur disquette avec l'instruction :

```
SAVE "PROG2"
```

Supprimez maintenant le programme en mémoire (NEW) et entrez maintenant le programme suivant, auquel nous fusionnerons le précédent :

```
10 REM programme 1
20 REM programme 1
30 REM programme 1
40 REM programme 1
50 REM programme 1
```

Nous pouvons maintenant entrer :

```
MERGE "PROG2"
```

L'ordinateur se comporte alors comme lors d'un chargement normal. Mais si vous entrez l'instruction LIST une fois le chargement terminé, vous obtenez :

```
10 REM programme 2
20 REM programme 1
25 REM programme 2
30 REM programme 2
40 REM programme 1
45 REM programme 2
50 REM programme 1
```

Vous voyez que les deux programmes ont été fusionnés. Les lignes 10 et 30 qui existaient dans les deux programmes sont conformes à celles du programme chargé en second. Une instruction qui vous sera donc très utile dans le développement de vos programmes, et qui vous permettra certainement de gagner beaucoup de temps.

Suppression de fichiers

Les programmes dont vous n'avez plus besoin ne doivent pas gaspiller éternellement la place mémoire, si précieuse, d'une disquette. Le DOS du CPC, AMSDOS, dispose d'une instruction de suppression des fichiers sur disquette. Toutes les instructions AMSDOS commencent par le caractère |. Le caractère | est dessiné sur votre clavier à côté de la touche P. On l'appelle souvent "barre verticale". Mais quand vous l'appuyez, cela affiche un "u accent grave", c'est-à-dire "ù" à l'écran. Dans toutes les commandes AMSDOS, vous pouvez donc taper ùCommande quand nous écrivons |Commande. Mais revenons à l'instruction elle-même :

Problème :	Suppression de fichiers
Instruction :	|ERA,"nom"

Paramètres :	"nom"	nom du fichier
Exemple :	|ERA,"secret.p" Supprime le programme "secret.p"	

Changer le nom de fichiers

Pour changer le nom d'un fichier, vous disposez de l'instruction AMSDOS suivante :

Problème :	Changement du nom d'un fichier	
Instruction :	|REN,"nouveau nom","ancien nom"	
Paramètres :	"nouveau nom" "ancien nom"	nouveau nom ancien nom
Exemple :	|REN,"DEMO.BAS","DEMO01.BAS" change le nom du programme DEMO01.BAS en DEMO.BAS	

Ici également, le suffixe du nom doit être également indiqué.

AMSDOS et surtout le système d'exploitation CP/M disposent certainement de bien d'autres instructions que nous n'avons pas évoquées dans ce chapitre. L'expérience prouve cependant que trop de bonnes choses risquent d'entraîner la confusion dans l'esprit du lecteur. C'est pourquoi nous nous limitons dans cet ouvrage aux instructions les plus importantes de gestion de programmes et fichiers sur disquette.

9. Encore plus d'instructions

Si vous êtes parvenu avec succès à ce chapitre du livre, vous serez certainement impatient de découvrir d'autres instructions du puissant BASIC du CPC. Vous allez être maintenant confronté à des instructions dont la complexité est limitée. Pour toutes les instructions, vous trouverez, comme à l'habitude, une description détaillée ainsi que des programmes d'exemple.

Interrogation du joystick

Vous pouvez connecter un joystick (ou manche à balai) sur le côté gauche de votre ordinateur. Les joysticks servent surtout dans les programmes de jeux. Voici l'instruction qui vous permet de connaître la position du joystick :

Problème :	Interrogation du joystick	
Instruction :	JOY (n)	
Paramètres :	n	numéro du joystick (0 ou 1)
Exemple :	PRINT JOY (0) Fournit le code correspondant à la position actuelle du joystick	
Remarque :	Les codes utilisés correspondent aux positions suivantes : 1 HAUT 2 BAS 4 GAUCHE 8 DROITE 9 HAUT-DROITE 10 BAS-DROITE 6 BAS-GAUCHE 5 HAUT-GAUCHE 32 FEU 1 16 FEU 2	

Voici maintenant un programme qui écrit sur l'écran les codes correspondant aux différentes directions que vous imprimez à votre joystick :

```
10 PRINT JOY(0)
20 GOTO 10
```

En fonction des valeurs que prend la variable système JOY (0), vous pouvez faire réagir un programme qui sera ainsi commandé par le joystick.

Les nombres aléatoires

Vous pouvez faire tirer des nombres au hasard (entre 0.000000001 et 0.99999999) par l'ordinateur avec l'instruction :

Problème :	Production de nombres aléatoires
Instruction :	RND (1)
Exemple :	PRINT RND (1) Produit un nombre aléatoire et le sort sur l'écran
Remarque :	Le nombre produit est compris entre 0.000000001 et 0.99999999

Une instruction très simple à utiliser. Les choses se compliquent cependant dès que vous voulez produire un nombre aléatoire appartenant à une autre fourchette. Si vous voulez par exemple produire un nombre entre 0 et 999, le nombre aléatoire doit être préparé comme dans le programme suivant :

```
10 nombre=RND(1)
20 PRINT INT(nombre*1000)
```

En effet, si nous multiplions le nombre aléatoire produit par la fonction RND par 1000, nous obtenons un nombre compris entre 0.000001 et 999.99999. La fonction INT supprime la partie non entière du nombre.

Si vous voulez cependant produire un nombre compris dans une fourchette dont la branche inférieure ne soit pas 0, il faut appliquer la formule suivante qui vous permet de produire des nombres aléatoires dans n'importe quelle fourchette de valeurs :

```
CONVERSION DE NOMBRES ALEATOIRES
INT (RND(1) * ((B+1)-A)))+A
A - plus petit nombre possible
B - plus grand nombre possible
```

Si vous voulez par exemple simuler des dés en produisant des nombres aléatoires compris entre 1 et 6, vous devez construire ainsi la formule qui convient :

```
INT (RND(1) * ((6+1)-1)))+1
```

qui donne :

```
INT (RND(1) * (7-1))+1
```

qui donne :

```
INT (RND(1) * 6)+1
```

Le programme suivant tire des nombres du Loto (6 nombres sur les nombres de 1 à 49) :

```
10 REM ******
20 REM  LOTO
30 REM ******
40 PRINT "Combien de series ";series
50 FOR i1=1 TO series
60 FOR i2=1 TO 6
70 PRINT INT(RND(1)*49)+1;" - ";
80 NEXT i2
90 PRINT
100 NEXT i1
```

Nous espérons que vous comprendrez aisément la logique de ce programme et nous allons même vous confier la réalisation d'une amélioration. Cette version du programme peut en effet

fort bien tirer deux fois le même nombre dans une même série. Essayez donc de faire un véritable programme de loto : placez les chiffres déjà tirés d'une série dans un tableau et n'acceptez les nombres tirés ensuite pour la même série que s'ils sont différents de tous les nombres du tableau. N'oubliez pas d'annuler tous les éléments de ce tableau avant de tirer la série suivante. Amusez-vous bien !

LEFT$

Les fonctions suivantes vous offrent de nouvelles possibilités en ce qui concerne le traitement des chaînes de caractéres. Ces fonctions sont très utiles et nous vous conseillons de bien en étudier la description.

Problème :	Isoler la partie gauche d'une chaîne	
Instruction :	LEFT$(chaîne,n)	
Paramètres :	chaîne	variable alphanumérique ou chaîne de caractères entre guillemets
	n	nombre de caractères devant être isolés (à partir du premier caractère de la chaîne)
Exemple :	PRINT LEFT$("ABCDEFG",3) Sort sur l'écran la partie composée des trois caractères les plus à gauche (ABC) de la chaîne "ABCDEFG"	
Remarque :	Si n est plus grand que le nombre de caractères effectif de la chaîne, la chaîne entière est reprise	

Cette fonction très intéressante vous permet donc de "détacher" la partie gauche d'une chaîne, comme le montre le programme suivant :

```
10 INPUT"Entrez votre nom";nom$
20 n=1
30 partie$=left$(nom$,n)
```

```
40 PRINT partie$
50 IF partie$=nom$ THEN 80
60 n=n+1
70 GOTO 30
80 PRINT"Fin"
```

Voici maintenant une instruction symétrique à LEFT$:

RIGHT$

Problème :	Isoler la partie droite d'une chaîne	
Instruction :	RIGHT$(chaîne,n)	
Paramètres :	chaîne	variable alphanumérique ou chaîne de caractères entre guillemets
	n	nombre de caractères devant être isolés (à partir du dernier caractère de la chaîne)
Exemple :	PRINT RIGHT$("ABCDEFG",3) Sort sur l'écran la partie composée des trois caractères les plus à droite (EFG) de la chaîne "ABCDEFG"	
Remarque :	Si n est plus grand que le nombre de caractères effectif de la chaîne, la chaîne entière est reprise	

Cette fonction ressemble tellement à la précédente que nous ne la décrirons pas plus en détail. Vous pouvez simplement modifier notre programme en remplaçant LEFT$ par RIGHT$. Vous pourrez ainsi voir nettement la différence entre les deux fonctions.

Voici maintenant une instruction qui appartient à la même famille que les deux précédentes mais qui vous permet d'isoler une partie de chaîne au milieu d'une chaîne de caractéres :

MID$

Problème :	Isoler une partie de chaîne	
Instruction :	MID$(chaîne,pos,n)	
Paramètres :	chaîne	variable alphanumérique ou chaîne de caractères entre guillemets
	pos	position du caractère à partir duquel doit commencer l'isolation de la partie de chaîne
	n	nombre de caractères devant être isolés (à partir du caractère de la chaîne situé en position pos)
Exemple :	PRINT MID$("ABCDEFG",2,4) Sort sur l'écran la partie composée des quatre caractères consécutifs en partant du second (BCDE) de la chaîne "ABCDEFG"	
Remarque :	Si n est plus grand que le nombre de caractères effectif de la chaîne, toute la partie restante de la chaîne est reprise	

Cette fonction permet donc d'isoler une partie quelconque d'une chaîne. Elle peut en fait aussi remplacer les deux instructions précédentes, mais elle est moins pratique dans ce cas puisqu'elle nécessite toujours un paramètre de plus. Ce paramètre précise en quelle position commence la portion de chaîne qu'il s'agit d'isoler.

Cette fonction est souvent utilisée pour sortir un par un les différents caractères d'une chaîne. Vous pouvez ainsi sortir par exemple les caractères d'une chaîne verticalement sur l'écran, comme dans le programme suivant :

```
10 a$="VERTICAL"
20 FOR p=1 TO 8:REM longueur de la chaine
30 PRINT MID$(a$,p,1)
40 NEXT p
```

La position du premier caractère à isoler est décalée de 1 sur la droite par la boucle p alors que la longueur de la portion de chaîne à isoler reste constante (1). Mais que faire si vous ne connaissez pas la longueur de la chaîne ? Bien sûr, il y a aussi une instruction pour déterminer la longueur de n'importe quelle chaîne :

Déterminer la longueur d'une chaîne

Problème :	Déterminer la longueur d'une chaîne	
Instruction :	LEN(chaîne)	
Paramètres :	chaîne	variable alphanumérique ou chaîne de caractères entre guillemets
Exemple :	A$="ABCDEFG":PRINT LEN(A$) Sort sur l'écran la longueur de la chaîne A$ (7)	
Remarque :	Si la chaîne est une chaîne vide, la fonction fournit le résultat 0	

Si nous appliquons maintenant cette fonction à notre dernier programme d'exemple, nous pouvons sortir verticalement n'importe quelle chaîne :

```
10 INPUT "Chaine:";a$
20 FOR p=1 TO LEN(a$)
30 PRINT MID$(a$,p,1)
40 NEXT p
```

Voici maintenant un petit programme assez spectaculaire qui vous montre un exemple d'application des fonctions de traitement des chaînes de caractères :

```
10 REM *****************
20 REM  MESSAGE DEFILANT
30 REM *****************
40 CLS
```

```
50 INPUT"Texte:";texte$
60 INPUT"Longueur:";longueur
70 INPUT"Vitesse:";vitesse
80 l=LEN(texte$)
90 texte$=STRING$(l,"-")+STRING$(l,"-")
100 FOR i=1 TO l
110 LOCATE 12,20
120 PRINT MID$(texte$,i,longueur)
130 FOR attendre=1 TO vitesse*50:NEXT attendre
140 NEXT i
150 GOTO 100
```

Le texte qui doit être montré en défilement est placé dans la variable texte$. La longueur du texte ainsi entré est conservée dans la variable 'l'. Pour que le texte soit entouré de signes de séparation, la ligne 90 a été introduite qui utilise l'instruction STRING$ que nous avons déjà employée dans la gestion d'adresses. La boucle suivante produit la portion de chaîne qui donne l'impression d'un défilement. La boucle d'attente en ligne 130 est parcourue entre 50 et 350 fois, suivant le nombre entré. Les deux boucles sont parcourues indéfiniment, tant que vous n'interrompez pas le programme avec la touche [ESC].

INSTR

Il est souvent nécessaire de rechercher si une chaîne contient une certaine suite de caractères, notamment dans les programmes de gestion de fichier. Voici une fonction qui facilite considérablement cette recherche :

Problème :	Recherche d'une chaîne dans une autre	
Instruction :	INSTR(pos,chaîne1,chaîne2)	
Paramètres :	pos	position à partir de laquelle doit commencer la recherche
	chaîne1	chaîne à l'intérieur de laquelle doit se faire la recherche
	chaîne2	chaîne recherchée

Exemple :	PRINT INSTR (3,"ABCDEFGHIJ","FGH") Recherche la chaîne "FGH" dans la chaîne "ABCDEFGHIJ" à partir de la troisième position de cette chaîne et sort sur l'écran la position de "FGH" (6) dans cette chaîne
Remarque :	Si la chaîne recherchée n'est pas trouvée, le résultat est 0. L'indication de la position est superflue si la recherche doit se faire dans la chaîne entière

Cette fonction est extrêmement utile pour rechercher des données dans un fichier. Voici un programme qui vous fournit un exemple de recherche d'une chaîne dans une autre :

```
5 CLS
10 INPUT "TEXTE:";texte$
20 INPUT "MODELE RECHERCHE:";cherche$
30 p=INSTR(texte$,cherche$)
40 IF p=0 THEN 70
50 PRINT"Le modele recherche se trouve en ";
55 PRINT "position";p;"du texte"
60 GOTO 80
70 PRINT"Le modele recherche n'a pas ete trouve"
80 PRINT"La recherche est terminee"
```

Ce programme vous permet donc d'entrer un texte puis un modèle à rechercher. La fonction INSTR fournit en ligne 30 le résultat de la recherche qui est placé dans la variable 'p'. Si la variable p n'est pas nulle, c'est que le modèle recherché a été trouvé. La position où il a été trouvé peut alors être indiquée.

Voici maintenant une version complétée de ce programme qui vous permet de faire remplacer le modèle recherché et trouvé par un autre texte :

```
5 CLS
10 INPUT "TEXTE:";texte$
20 INPUT "MODELE RECHERCHE:";cherche$
30 INPUT "REMPLACER PAR:";remplace$
```

```
40 p=INSTR(texte$,cherche$)
50 IF p=0 THEN 90
60 texte$=LEFT$(texte$,p-1)+remplace$+
   MID$(texte$,p+LEN(cherche$),LEN(texte$)-
   LEN(cherche$)-(p-1))
70 PRINT"Nouveau texte: ";texte$
80 GOTO 100
90 PRINT"Le modele recherche n'a pas ete trouve"
100 PRINT"La recherche est terminee"
```

Ne vous laissez pas impressionner par la ligne 60 et examinez calmement les différents éléments qu'elle réunit pour former le nouveau texte :

1. La partie de l'ancien texte située avant le modèle trouvé (p-1)

2. Le texte de remplacement (remplace$)

3. La partie de l'ancien texte située aprés le modèle trouvé (de p+LEN(cherche$) à la fin du texte en LEN(texte$)-LEN(cherche$)-(p-1))

Vous rencontrerez de telles routines de recherche et de remplacement dans pratiquement tous les programmes de traitement de texte. Le CPC 6128 Plus vous permet de réaliser une telle routine de la façon la plus simple.

Annexe 1 - Les mots réservés

Les noms que vous donnez aux variables peuvent être choisis librement, avec cette seule réserve très importante que le nom d'une variable ne doit jamais être identique à un mot réservé, c'est-à-dire à un mot qui fait partie du vocabulaire de votre CPC 6128 Plus, soit qu'il s'agisse d'un mot-instruction, soit qu'il s'agisse du nom d'une variable du système. Par contre, vous pouvez parfaitement choisir des noms de variables constitués d'un mot réservé et d'autres caractères. DATA ne peut par exemple pas être employé comme nom de variable mais DATA1 peut parfaitement désigner une variable. Si vous négligez cette régle, vous serez rappelé à l'ordre par un message 'Syntax error'. Si vous ne comprenez pas la cause d'un message 'Syntax error', vérifiez dans la liste suivante si vous n'avez pas utilisé indûment un mot réservé.

ABS
AFTER
AND
ASC
ATN
AUTO
BIN$
BORDER
BREAK

CALL
CAT
CHAIN
CHAIN MERGE
CHR$
CINT
CLEAR
CLEAR INPUT
CLG

CLOSEIN
CLOSEOUT
CLS
CONT
COPYCHR$
COS
CREAL
CURSOR
DATA
DEC$

DEF FN
DEFINT
DEFREAL
DEFSTR
DEG
DELETE
DERR
DI
DIM
DRAW
DRAWR
EDIT
EI
ELSE
END
ENT
ENV
EOF
ERASE
ERL
ERR
ERROR
EVERY
EXP
FILL
FIX
FN
FOR
FRAME
FREE
GOSUB
GOTO
GRAPHICS PAPER
GRAPHICS PEN
HEX$
HIMEM
IF
INK
INKEY
INKEY$
INP
INPUT
INSTR
INT
JOY
KEY
KEY DEF
LEFT$
LEN
LET
LINE INPUT
LIST
LOAD
LOCATE
LOG
LOG10
LOWER$
MASK
MAX
MEMORY
MERGE
MID$

MIN

MOD

MODE

MOVE

MOVER

NEW

NEXT

NOT

ON BREAK CONT

ON BREAK GOSUB

ON BREAK STOP

ON ERROR GOTO

ON... GOSUB

ON... GOTO

ON SQ GOSUB

OPENIN

OPENOUT

OR

ORIGIN

OUT

PAPER

PEEK

PEN

PI

PLOT

PLOTR

POKE

POS

PRINT

PRINT SPC

PRINT TAB

PRINT USING

RAD

RANDOMIZE

READ

RELEASE

REM

REMAIN

RENUM

RESTORE

RESUME

RESUME NEXT

RETURN

RIGHT$

RND

ROUND

RUN

SAVE

SGN

SIN

SOUND

SPACE$

SPC

SPEED INK

SPEED KEY

SPEED WRITE

SQ

SQR

STEP

STOP

STR$

STRING$

SWAP

SYMBOL

SYMBOL AFTER

TAG

TAG OFF

TAN

TEST

TESTR

THEN

TIME

TO

TROFF

TRON

UNT

UPPER$

USING

VAL

VPOS

WAIT

WEND

WHILE

WIDTH

WINDOW

WINDOW SWAP

WRITE

XOR

XPOS

YPOS

ZONE

Annexe 2 - Les messages d'erreur

Les messages d'erreur sont tout aussi importants que les instructions d'un ordinateur. Plus un ordinateur dispose de messages d'erreur différents, plus il est aisé pour l'utilisateur de repérer ses erreurs. Imaginez en effet dans quelle perplexité vous seriez plongé si votre ordinateur réagissait à toute erreur en vous envoyant le message 'Syntax error' ! Heureusement, le CPC 6128 Plus dispose d'un bon nombre de messages d'erreur que nous décrivons ci-dessous. Vous pourrez donc consulter cette liste chaque fois que vous rencontrerez un message d'erreur que vous ne connaissez pas encore.

Le numéro d'erreur que nous avons placé devant chaque message d'erreur est nécessaire à la détection des erreurs en mode programme (ON ERROR GOTO). Cette instruction n'ayant pas été abordée dans cet ouvrage, ce n'est que plus tard que vous aurez l'usage du numéro d'erreur.

1 Unexpected NEXT

L'instruction NEXT a été rencontrée dans le programme sans qu'une boucle ait été ouverte auparavant ou la boucle ouverte auparavant l'a été avec une autre variable que celle utilisée dans l'instruction NEXT.

2 Syntax error

Vous connaissez déjà ce message d'erreur. Il signifie que le CPC 6128 Plus ne comprend pas la ligne d'instructions que vous avez entrée.

3 Unexpected Return

L'instruction Return a été rencontrée dans le programme sans qu'une instruction GOSUB ait auparavant appelé un sous-programme.

4 DATA exhausted

Vous avez tenté de lire avec READ plus de données en lignes DATA qu'il n'y en a.

5 Improper argument

Vous avez utilisé une fonction ou un paramètre non valable.

6 Overflow

Le résultat d'un calcul dépasse le nombre le plus grand que peut stocker l'ordinateur ou ne peut être codé sous la forme d'un nombre 16 bits pour une variable entière.

7 Memory full

Le programme ou les variables dépassent la capacité mémoire disponible. L'instruction MEMORY provoque ce message d'erreur lorsque vous tentez de placer le début du Basic dans un endroit incorrect.

8 Line does not exist

Un numéro de ligne fourni ne correspond pas à une ligne existant dans le programme.

9 Subscript out of range

L'indice utilisé pour un tableau est supérieur à l'indice maximum réservé avec l'instruction DIM.

10 Array already dimensioned

Vous avez tenté de dimensionner un tableau dans une instruction DIM alors qu'il avait déjà été dimensionné.

11 Division by zero

Vous avez tenté d'effectuer une division par 0, ce qui n'est pas possible en mathématiques.

12 Invalid direct command

Vous avez entré une commande en mode direct qui ne peut être utilisée que dans les programmes.

13 Type mismatch

Vous avez fourni une valeur numérique là où il fallait fournir une chaîne ou l'inverse.

14 String space full

Il n'y a plus de place en mémoire pour stocker d'autres chaînes de caractéres.

15 String too long

La taille limite d'une chaîne (255 caractères) a été dépassée, par exemple lors d'une opération de concaténation.

16 String expresssion too long

La taille limite d'une chaîne (255 caractères) a été dépassée lors du stockage provisoire que l'ordinateur effectue pour toute opération sur des chaînes de caractères.

17 Cannot CONTinue

Le programme ne peut être poursuivi avec l'instruction CONT.

18 Unknown user function

Une fonction appelée avec FN n'a pas été définie auparavant avec DEF FN.

19 RESUME missing

Le programme a été interrompu dans une routine ON ERROR GOTO.

20 Unexpected RESUME

Une instruction RESUME ne peut être exécutée car aucune routine ON ERROR GOTO n'a été appelée.

21 Direct command found

Lors du chargement d'un programme sur cassette, une ligne sans numéro de ligne a été rencontrée.

22 Operand missing

Un paramètre indispensable n'a pas été fourni.

23 Line too long

Une ligne Basic dépasse la taille maximale de 255 caractères.

24 EOF met

Vous avez tenté de lire plus de données sur un fichier qu'il n'en contient.

25 File type error

Le type de fichier ne correspond pas à l'instruction de lecture utilisée. Par exemple, vous ne pouvez pas charger un fichier avec LOAD.

26 NEXT missing

Une boucle FOR a été programmée sans instruction NEXT correspondante.

27 File already open

Vous avez tenté d'ouvrir un fichier qui n'a pas encore été fermé.

28 Unknown command

Vous avez entré une instruction qui ne fait pas partie du jeu d'instructions de l'ordinateur.

29 WEND missing

Une boucle WHILE a été programmée sans instruction WEND correspondante.

30 Unexpected WEND

Une instruction WEND a été rencontrée sans qu'une instruction WHILE n'ait été trouvée auparavant.

31 File not open

Vous avez essayé de lire ou d'écrire dans un fichier sans l'ouvrir au préalable (OPENIN, OPENOUT).

Notes

Notes

Notes

Notes

Notes

Notes

Achevé d'imprimer
sur les presses de l'imprimerie IBP
à Rungis (Val-de-Marne 94) (1) 46.86.73.54
Dépôt légal - Octobre 1990
N° d'impression: 5383

AMSTRAD

6128 plus

BIEN DEBUTER

SZCZEPANOWSKI

L' Amstrad 6128 plus: une console de jeux mais surtout un ordinateur doté d'un langage de programmation complet et rapide, le Locomotive Basic, et d'un logiciel d'exploitation, le CP/M. Grâce à ces deux outils, vous pourrez copier ou formater une disquette, lancer un jeu et réaliser vos premiers programmes. Avec BIEN DEBUTER 6128 PLUS vous apprendrez à utiliser l'écran, le clavier, le lecteur de disquettes ainsi qu'à dessiner et composer de la musique.

Principaux sujets traités:

- Utilisation du clavier et des touches spéciales.
- Les instructions: afficher un message, réaliser des calculs...
- Le langage Basic: fichier d'adresses, tableaux, boucles...
- Couleur et graphisme: couleurs, points, cercles et droites...
- Le son: hauteur d'une note, fréquences musicales...
- Le lecteur de disquettes: formater une disquette, lui donner un nom...
- Le joystick, les nombres au hasard...



Réf. ML 759. Prix 78 F.
ISBN : 2-86899-393-1 / ISSN : 0980-1928

EDITIONS MICRO APPLICATION
58 RUE DU FAUBOURG POISSONNIERE
75010 PARIS TEL (1) 47 70 32 44

ML 759

MA

BIEN DEBUTER

AMSTRAD 6128plus



Document numérisé avec amour par

AMSTRAD CPC

MÉMOIRE ÉCRITE



https://acpc.me/